University of Zurich[UZH]

Department of Informatics

Learning from the Octopus: Sensorimotor Control of Octopus-Inspired Soft Robots

A dissertation submitted to the Faculty of Economics, Business Administration and Information Technology
of the University of Zurich

for the degree of
Doctor of Science (Ph.D.)

by

Tao Li

from China

Accepted on the recommendation of

Prof. Dr. Rolf Pfeifer
Prof. Akio Ishiguro, Ph.D.

2013

The Faculty of Economics, Business Administration and Information Technology of the University of Zurich herewith permits the publication of the aforementioned dissertation without expressing any opinion on the views contained therein.

Zurich, October 23, 2013

Head of the Ph.D. committee for informatics: Prof. Abraham Bernstein, Ph.D

Abstract

Soft robotics is one of the most promising, yet challenging, research topics in bio-inspired robotics. In terms of morphological and behavioral flexibility as well as interaction safety, soft robots have significant advantages over traditional articulated rigid robots. However, it is difficult to achieve autonomous sensorimotor control of a soft robot by conventional engineering approaches primarily because of its complex nonlinear soft body dynamics. Most soft robots are currently controlled only in an open-loop scheme.

The octopus exhibits various sophisticated behaviors with muscular-hydrostats and apparently has solved the challenges to harness a completely soft body. Studying the control strategy of the octopus, by constructing robots inspired by the octopus' morphology and implementing octopus-like behaviors, can provide insights into understanding its biological control scheme and designing novel control methods for soft robots.

Many researchers have used global parameters to reduce the control dimensions of soft robots, but that is not sufficient for autonomous sensorimotor control. In this thesis, we investigate the sensorimotor control strategies of octopus-inspired soft robots for both single-arm manipulation and multi-arm coordination tasks. By reviewing and interpreting biological studies on the octopus, we propose a control scheme to implement autonomous behaviors on soft robots. Several octopus arm simulators and octopus-inspired soft robotic platforms are built to evaluate the control scheme. We evaluated the control scheme by: (1) controlling a simulated octopus arm for a reaching task; (2) embedding and switching among multiple behaviors for a single soft robotic arm; and (3) achieving autonomous direction and speed control of a multi-arm soft robot with the coordination of all arms. We found that initiating these global parameters at the proper time is a key factor to achieving autonomous sensorimotor control in soft robots. Finally, we show the potential to exploit complex soft body dynamics as a computational resource.

With this work, we present a promising approach both for the control of soft robots and for morphological computation.

Zusammenfassung

Soft-Robotik ist eine der vielversprechendsten, jedoch auch herausforderndsten Forschungsgebiete innerhalb der biologisch inspirierten Robotik. Im Gegensatz zu traditionellen Robotern, welche starre Strukturen aufweisen und aus harten Materialien aufgebaut sind, haben Soft-Roboter hinsichtlich der Flexibilität der Morphologie und der Bewegungen einen erheblichen Vorteil. Dazu kommt die erhöhte Interaktionssicherheit, die Soft-Roboter aufgrund ihrer elastischen Materialeigenschaften mit sich bringen. Es ist jedoch sehr schwierig, mit den gängigen Methoden ein autonomes, sensomotorisches Regelsystem für diese Roboter zu entwickeln. Grund dafür ist das komplexe, nichtlineare Verhalten elastischer Körper. Demzufolge werden die meisten Soft-Roboter heutzutage nur mittels eines einfachen offenen Regelkreises gesteuert.

Der Oktopus zeigt mit seiner speziellen muskulären Struktur (muscular hydrostat) hoch komplexe Verhaltensweisen und schafft es offensichtlich, seinen vollständig elastischen Körper ohne Probleme zu kontrollieren. Durch das Entwickeln von Oktopus-inspirierten Robotern, welche ähnliche Körpereigenschaften und Verhaltensweisen aufweisen, versuchen wir besser zu verstehen, wie das Tier seine Bewegungen steuert. Die Erkenntnisse dieser Forschung können dann in Folge zur Entwicklung neuer Steuerungssysteme für Soft-Roboter beitragen.

Viele Forscher haben versucht, die Dimensionen der Steuerungssysteme von Soft-Robotern mit globalen Parametern zu reduzieren. Dies ist jedoch bei autonomen, sensomotorischen Regelsystemen nicht ausreichend. In dieser Dissertation untersuchen wir die sensomotorischen Steuerungssysteme von Oktopus-inspirierten Soft-Robotern, insbesondere bei der Manipulation mit einem Arm sowie bei der Koordination von mehreren Armen. Basierend auf biologischen Studien des Oktopus entwickelten wir ein Regelsystem für die Implementierung von autonomen Verhaltensweisen bei Soft-Robotern. Um das Regelsystem zu evaluieren, entwickelten wir sowohl mehrere Simulationen von Oktopusarmen als auch mehrere Oktopus-inspirierte Soft-Roboter-Plattformen. Folgende Experimente dienten der Evaluation: (1) Die Steuerung einer Greifbewegung mit einem simulierten Oktopusarm, (2) das Einbinden und Wechseln von Bewegungsabläufen in einem echten, einarmigen Soft-Roboter und (3) das Erzielen einer autonomen Richtungssteuerung und Kraftregelung in einem mehrarmigen Soft-Roboter unter Berücksichtigung der Koordination aller Arme. Wir beobachteten, dass bei der Initiierung der globalen Parameter der richtigen Zeitpunkt ausschlaggebend ist, um eine autonome, sensomotorische Steuerung bei Soft-

Robotern zu erzielen. Schliesslich zeigten wir das Potential, das in der Verwendung der komplexen Dynamik von Soft-Robotern als Rechenressource (computational resource) liegt.

Mit dieser Arbeit präsentieren wir einen vielversprechenden Ansatz für die Steuerung von Soft-Robotern und leisten einen Beitrag im Gebiet der Morphological Computation.

Acknowledgements

First of all, I would like to thank my thesis supervisor Rolf Pfeifer to provide the valuable opportunity to carry out my research at the legend Artificial Intelligence Laboratory (AI Lab). I wish to thank my thesis co-supervisor Akio Ishiguro for giving thoughtful and constructive feedbacks. During the expedition of my Ph.D. study, I have received enormous help and continuous encouragement from Kohei Nakajima. I sincerely appreciate all the inspiring discussions and sharing of research experiences. Without his help, the journey would be much more difficult. Many thanks to Nathan Labhart and Qian Zhao for kindly spending time to proofread my thesis draft. The thesis abstract was translated into German by Dorit Assaf, Helmut Hauser, and Nathan Labhart. Thanks all for your help. I would like to acknowledge all my colleagues at the AI Lab for fruitful discussions and shared time.

This research was mainly funded by the European Commission under the Seventh Framework Programme for Research and Technological Development (FP7) in the theme of the Future and Emerging Technologies (FET). It was conducted as part of the ICT-FET OCTOPUS Integrating project (EU project FP7-231608). I am very appreciated for the four and half years support. I wish to express my gratitude for my project collaborators who have contributed to the studies included in this thesis.

Living and studying in a foreign country is both exciting and challenging. Thanks Nathan Labhart, Dorit Assaf, Lijin Aryananda, Alejandro Hernandez Arieta, and many others who have helped me to deal with lots of complicated situations and to settle down in Switzerland. Thanks all my friends to make life more enjoyable and to assist me in various occasions.

I own a lot to my family: my parents, Zhenlan Niu and Yuchun Li, are always my firm supporters and have provided all the resources they own for my development and education; my older sister and her family (Juan Li, Mingwei Li, Jiaqi Li, and Jiaze Li) have always given me the sincere love and thoughtful care; my wife, Ni Jiang, has accompanied and supported me all the way from China, Singapore to Switzerland; My son, Ruisheng Li, brings me lots of fun and smile. Thank you all!

Contents

List of Figures

Chapter 1

Introduction and Motivation

It is widely recognized that the behaviors of embodied systems emerge from the reciprocal interactions among the control, the body, and the environment [Pfeifer et al., 2007]. Sensorimotor coordination, the mutual coupling of sensing and acting, is essential to behavior formation. By incorporating sensory modality, which is essential but commonly missing in soft robots built so far, we investigate the sensorimotor control of octopus-inspired soft robots, as well as its implications for the octopus biology and soft robots in general.

1.1 Emergence of Soft Robotics

In a traditional rigid-link robot, the end effector position is fully determined by its link lengths and joint angles. These kind of robots, especially industrial robotic manipulators, are good at accomplishing precise repetitive tasks in a structured workspace where the environment can be accurately modeled. However, these rigid robots have a limited tolerance for the changes in its working environment. In contrast to the prevalence of rigid articulated structures in traditional robots, biological systems have more diverse constructions and routinely deal with uncertainty in their surroundings. One interesting observation is that soft elements are ubiquitous in the construction of biological systems. The extreme are animals that have completely soft bodies, without any rigid skeleton. Instead, they use highly compliant materials and flexibly vary their stiffness using hydraulics, muscle tension, and tissue compaction [Kier and Smith, 1985]. During the last two decades, roboticists have been investigating the opportunities and challenges of hyper-redundant robots, continuum robots, and recently, soft robots. Figure 1.1 illustrates the spectrum of bio-inspired robots in terms of flexibility and state controllability. We can see that as the softness and flexibility increases, it is getting more and more difficult to control the robots. More effort should be devoted to address the control challenges of soft robots.

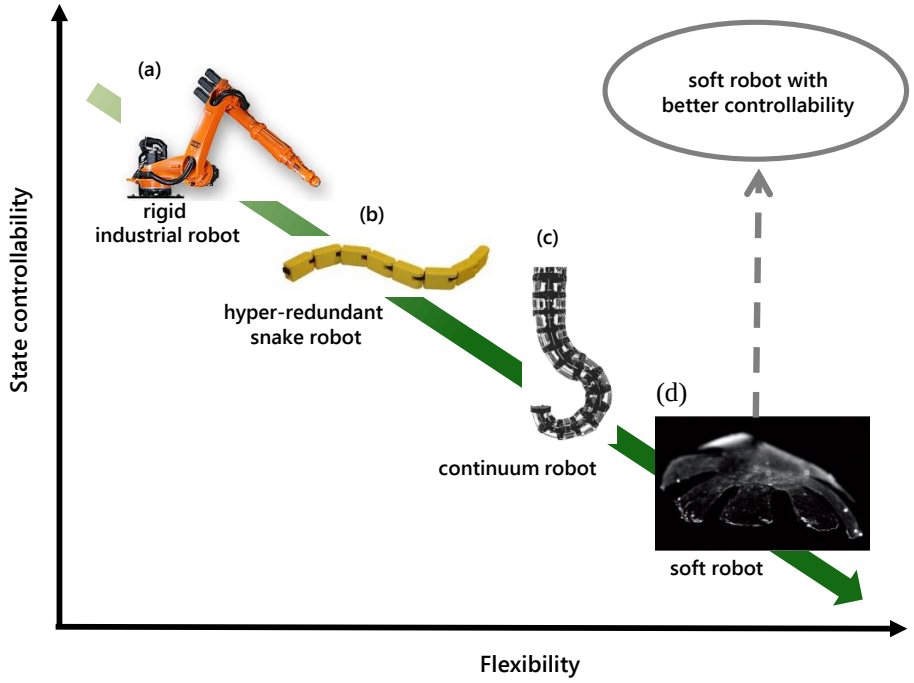


Figure 1.1: Spectrum of bio-inspired robots in terms of flexibility and state controllability. (a) A KUKA industrial robot [Swevers et al., 2007]; (b) AmphiBot: an amphibious snake robot [Ijspeert and Crespi, 2007]; (c) a continuum manipulator inspired by the elephant's trunk [Hannan and Walker, 2003]; (d) a synthetic jellyfish [Nawroth et al., 2012]. As the increase of softness and flexibility, the control challenges also intensify.

1.1.1 Softness and Soft Robotics

Softness is a commonly used meanwhile ambiguous term, which is often confused for elasticity or flexibility. Firstly, elasticity and flexibility are closely related but different. Both are measures of the tendency to return to its original shape during elastic deformation. The former is an intensive property of the material and is normally quantified by elastic modulus; while the later, or its opposite stiffness, is an extensive property of a structure and dependents on the constituent material, the shape, and boundary condition [Beer et al., 2009; Sadd, 2009]. The stiffness of a structure is proportional to the tensile elastic modulus of its constituent material. Secondly, softness, or its more commonly used opposite hardness, is another intensive property of a material and is a measure of how resistant a material is to permanent shape change due to applied forces. As a result, although structures made of soft materials tend to be flexible, but softness and flexibility are not directly related to each other. For example, steel is not a soft material but a thin and narrow steel plate with support at one end is a flexible structure; rubber is a soft material but it can be used to create structures such as wheels, which are not flexible in normal working condition.

Soft robotics represents one of the new frontiers and trends in robotics research [Iida and Laschi, 2011; Pfeifer et al., 2012; Kim et al., 2013; Majidi, 2013]. Figure 1.2 illustrates the dramatically in-

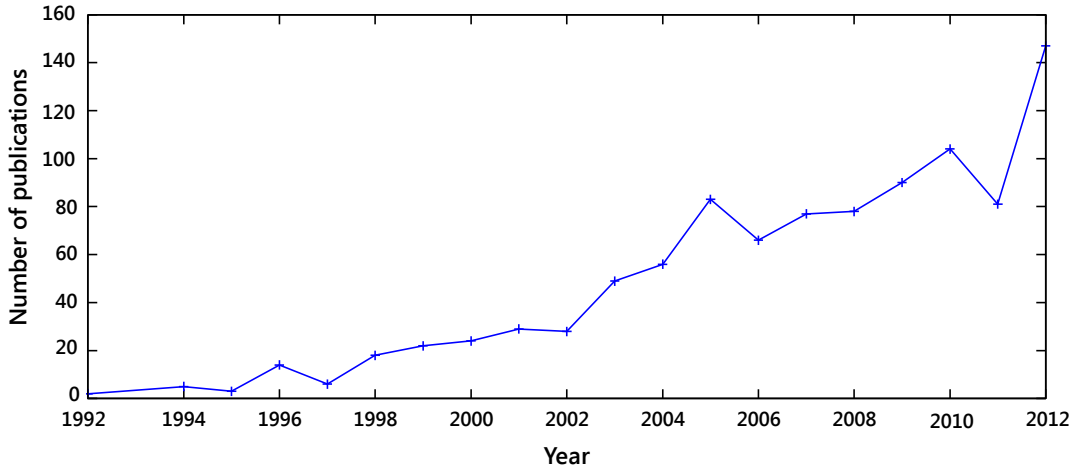


Figure 1.2: Increasing number of scientific publication in the scope of soft robotics from 1992 to 2012 [Thomson Reuters, 2013].

creased number of scientific publications in the scope of soft robotics from 1992 to 2012, especially in the last ten years (2002–2012). We notice that the term *soft robotics* has been used in a number of different contexts in robotics literature [Filippini et al., 2008; Albu-Schaeffer et al., 2008; Trivedi et al., 2008; Albu-Schaeffer et al., 2009; Calisti et al., 2011; Kim et al., 2013; Majidi, 2013]. In this thesis, we use this term to refer to the study on construction and control of robots made of soft materials such as colloids, emulsions, drops, polymers and gels. This is different from using the same term for robots with features of soft robots, such as torque-controlled flexible joints or variable compliance actuation [Albu-Schaeffer et al., 2009]. We consider hyper-redundant robots [Trivedi et al., 2008] as an intermediate step between traditional rigid robots and soft robots. We compare the differences between soft and hyper-redundant robots later in this chapter.

A soft robot constructed by soft materials, for instance silicone rubber, exhibits large deformations under normal operation conditions. In terms of morphological flexibility and interaction safety, they have significant advantages over traditional articulated robots, which resemble the limb structures of vertebrates or arthropods and are constructed mainly by rigid engineering materials, such as metal and plastic. The flexibility gives soft robots the potential to be used as, for example, search and rescue robots which could crawl through rubble and squeeze into constrained spaces. Another key attribute of soft robots, which is not easy to achieve with conventional robots, is whole-arm manipulation. In this mode of operation, a soft robot grasps and manipulates objects by curling part of its structure around the object, adapting to the object's shape, size, and the dynamic properties of the environment. The arm essentially becomes its own highly adaptable end effector.

1.1.2 Soft Robot Designs

In recent years, several research groups have attempted to build soft robots with the capability to deform [Brown et al., 2010; Ilievski et al., 2011], crawl [Shepherd et al., 2011; Lin et al., 2011], swim [Otake et al., 2002; Suzumori et al., 2007; Nawroth et al., 2012], jump [Sugiyama and Hirai, 2006], or other locomotion modes [Steltz et al., 2009; Umedachi et al., 2010; Onal and Rus, 2013]. Figure 1.3 presents some representative design endeavors. Apparently, soft robots have quite diverse morphologies and actuation methods, as a result the designs cannot be easily transferred from one robot to another.

The design objectives and functionalities of these soft robots can be classified into two categories:

1. To mimic and study the flexible locomotion behaviors of animals, including but not limited to starfish [Otake et al., 2002], manta [Suzumori et al., 2007], true slime mold [Umedachi et al., 2010], caterpillar [Lin et al., 2011], jellyfish [Nawroth et al., 2012], and snake [Onal and Rus, 2013].

2. To achieve manipulation by adaptively conforming the soft robot body to objects. A universal robotic gripper [Brown et al., 2010] and a robotic manipulator to handle fragile objects [Ilievski et al., 2011] are two examples of this category.

Soft robots studied in this thesis are different with hyper-redundant robots, which have also been referred as soft robots in literature. In what follows, we draw a comparison between soft robots and hyper-redundant robots, especially its subcategory continuum robots.

1.1.3 Differences between Soft and Hyper-Redundant Robots

The degrees of freedom (DoFs) of a robot equals to the number of independent variables required to describe the robot's state and to determine the possible robotic configurations. Traditional robot manipulators have a relatively limited number of DoFs, seven or less [Crane, 2008]. However, to operate in a complex, cluttered environment or to perform flexible and versatile manipulation tasks, a traditional manipulator with a small number of DoFs will not likely to perform adequately. Manipulators designed with high DoFs to achieve enhanced flexibility are commonly referred as hyper-redundant manipulators [Chirikjian and Burdick, 1991]. A previous study [Chirikjian, 1992; Chirikjian and Burdick, 1994] presented an excellent classification and abstraction of hyper-redundant and continuum robot morphologies, illustrated in Figure 1.4. Three categories were identified: (a) discrete morphology, (b) continuous morphology, and (c) cascade of independent modules. Examples of the discrete morphology are snake robots [Dowling, 1997; Hopkins et al., 2009] and snake-like hyper-redundant manipulators [Mochiyama et al., 1998]. Hyper-redundant robots with continuous morphologies are also

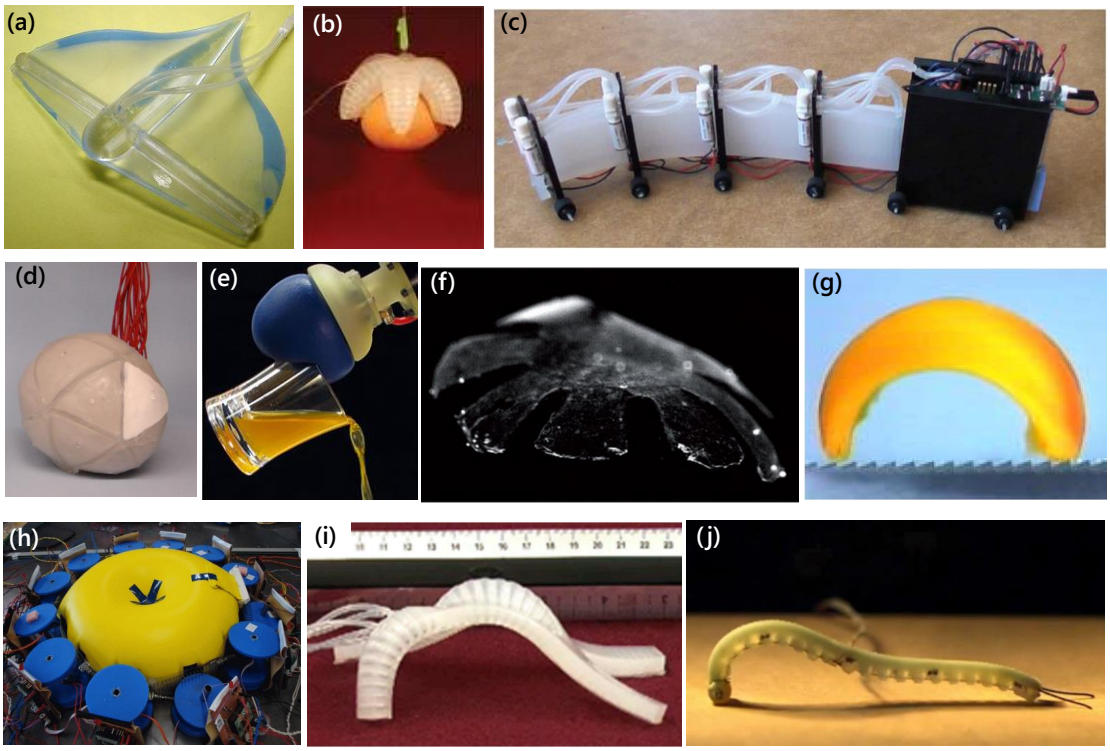


Figure 1.3: Recent soft robots. (a) A soft swimming robot inspired by a manta ray and composed of only rubber material. It moves smoothly in water by two embedded bending pneumatic rubber actuators [Suzumori et al., 2007]. (b) A soft robotic gripper holding and lifting a fragile raw egg by pneumatic driven mechanism, the gripper consists of a starfish-like elastomer structure with embedded pneumatic networks [Ilievski et al., 2011]. (c) A soft robotic snake with on-board actuation, power, and control abilities. The robot consists of four serially connected bidirectional fluidic elastomer actuators [Onal and Rus, 2013]. (d) A soft mobile robot capable of rolling over by shape deformation. It is pneumatically actuated by the transition of particulate materials between a liquid-like and a solid-like state in its surface and an internal expanding actuator in its center [Steltz et al., 2009]. (e) A robotic gripper based on the jamming of granular materials. It is able to conform to and pick up objects of various shape and surface properties. The hardness of granular materials filled in the blue flexible sac is controlled by vacuum pressure [Brown et al., 2010]. (f) A synthetic jellyfish constructed from silicone polymer covered by a monolayer of rat heart tissue. It is actuated by an external electrical field stimulator. The motion of the synthetic medusoid closely resembles that of the propulsion and feeding of a jellyfish [Nawroth et al., 2012]. (g) A biomimetic gel robot that can form a worm-like motion by bending and stretching without external driving stimuli. It converts an oscillating chemical reaction into kinetic energy to generate motions [Maeda et al., 2007]. (h) A soft robot that exhibits amoeboid locomotion inspired by a true slime mold [Umedachi et al., 2011]. (i) A soft robot fabricated by soft lithography capable of sophisticated locomotion. It is actuated by applying low-pressure air to a series of chambers embedded in a layer of elastomer, which is bonded to another relatively inextensible and compliant sheet [Shepherd et al., 2011]. (j) A completely soft robot built with the inspiration of a caterpillar. The robot is constructed from highly elastic silicone rubber. It moves by using shape memory alloy springs as actuators [Lin et al., 2011].

called continuum robots, which are usually constructed with serially connected pneumatic actuators [Suzumori et al., 1991; Jones and Walker, 2006] or tendon-driven sections separated by rigid plates [Hannan and Walker, 2003]. Variable geometry truss manipulators [Hughes et al., 1991] belong to the third category as well.

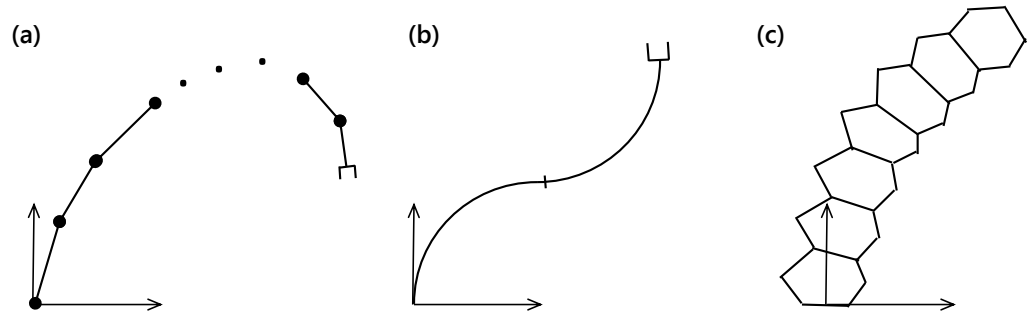


Figure 1.4: Three categories of hyper-redundant robot morphologies (adapted from [Chirikjian, 1992]). Type (a) has a discrete morphology and is built with a large number of relatively short rigid links. Type (b) is also called continuum robot. It is made by flexible links, so the deformation is distributed continuously over each link length. Type (c) is constructed by a cascade of modules actuated in parallel.

Although soft robots also have a large number of DoFs, they are different from hyper-redundant and continuum robots in morphology and in operating approaches. Soft robots use elastic materials extensively or exclusively; in contrast, to achieve precision, type (a) and (c) of hyper-redundant robots rarely adopt elastic materials. Continuum robots partially use flexible or elastic materials and are developed using mechanisms based on traditional robotics, so they have rigid elements (back bones or endplates) that provide supporting points to exert forces. In terms of operation, hyper-redundant and continuum robots aim to precisely control their end effectors using kinematics and dynamics-based methods, or by approximating the geometry of their back bones. Instead of precise control of end effectors, soft robots aim at achieving functionalities with their full body deformation. Nevertheless, the distinction between soft robots and hyper-redundant robots is not conspicuous.

After examining the characteristics of soft robots by surveying representative soft robot designs in the literature and summarizing their differences from hyper-redundant robots, we investigate the challenges of controlling soft robots, specifically closed-loop sensorimotor control, in the next section.

1.2 Challenges to Achieve Sensorimotor Control of Soft Robots

Along with the promising potentials and advantages over traditional rigid robots, there are enormous challenges and unique issues to achieve sensorimotor closed-loop control of soft robots:

- Soft materials exhibit highly nonlinear and time-varying dynamics under actuation [Meier et al., 2005]. These materials exhibit both elastic and viscous responses; therefore, they are

called viscoelastic materials. The effects of the applied forces propagate in a soft structure in an unpredictable manner. It is essential to track and embed the varying dynamics of the robot and the environment using, for example, machine learning techniques.

- The whole structure of a soft robot usually has large elastic deformations and shows geometrical nonlinearity under normal working conditions. This property prevents adopting traditional robotic control methods based on rigid-structure kinematics and dynamics because the traditional method is based on small deformation and even transformation hypothesis [MacKerrow, 1995].
- Soft robots contain unlimited DoFs. It is impractical to provide and control a correspondingly unlimited number of independent actuators. As a result, soft robots are intrinsically under-actuated systems [Tedrake, 2009].
- It is not a trivial task to equip soft robots with sensors, which are indispensable to implement closed-loop sensorimotor control. There are limited choices of commercially available sensors which can be embedded into soft robots, and embedding sensors and their power and signal cables will reduce the flexibility of soft robots.

As a result of these challenges, sensorimotor control is normally missing in the soft robotics literature. Nevertheless, we can summarize some common and beneficial control principles for controlling soft robots.

1.3 Previous Attempts to Control Soft Robots

The soft robotics literature has primarily focused on structure and system design, while control issues are rarely treated as the main focus. We extract two control principles, which are generally applicable in controlling soft robots.

Reduce Control Complexity by Using Morphology and Material Properties

It is well known that the behavior of a robotic system is not only the result of internal control but is shaped by the interaction between its control, body, and the environment [Pfeifer and Scheier, 1999; Pfeifer and Bongard, 2006; Pfeifer et al., 2007]. It is clear that the control overhead of soft robots can be significantly reduced by the use of soft materials and an appropriate design of the robot morphology. A previous study [Onal and Rus, 2013] presented a fluidic soft snake robot (Figure 1.3(c)) which exploited slow soft body dynamics to convert an electrical square wave input to a mechanical sinusoidal output. As a result, smooth undulatory serpentine locomotion was generated by controlling only the binary operations of solenoids without the need for complicated control systems. In case of the universal robotic gripper (Figure 1.3(e)), the need to position the gripper precisely and to reconfigure for different types of objects is no longer required because

of the properties of granular materials under vacuum pressure. Experiments on a synthetic jellyfish (Figure 1.3(f)) demonstrate the critical role of body geometry and material arrangement. By appropriate design of the distribution and size of embedded pneumatic chambers, a soft robot (Figure 1.3(i)) produces diverse gaits using only a single actuation source of air pressure.

Reduce Control Complexity by Using Global Parameters and Self-Organization

Because of the unlimited DoFs of soft robots, it is undesirable and impractical to control each DoF individually. Meanwhile, interesting movements can be self-organized through the local interaction among different robot modules and/or the interaction between the robot body and the environment with only limited control input. Global commands are commonly used as the trigger for the self-organizing emergent process. Because the number of global commands needed is usually limited and much smaller, the control dimension and complexity are reduced. The universal gripper, shown in Figure 1.3(e), changes between fluidic and solid states and adaptively holds a wide range of objects with only one simple global command: the on/off of a vacuum. For the synthetic jellyfish called medusoid (Figure 1.3(f)), the global commands employed to generate the free swimming behavior are the strength and phase of pulse stimulation from an external electrical field. Work by Umedachi and coauthors [Umedachi et al., 2010] introduced a way to implement sensorimotor control in a slime-mold-inspired soft robot through decentralized control. The robot achieves amoeboid locomotion using local sensory feedback mechanism and behavior self-organization, without global planning.

To summarize, although there are some implicit or explicit control principles, it is profound that many soft robots are designed and implemented by intuition and empirical experimentation [Ilievski et al., 2011]. This approach is useful in demonstrating soft robotic behaviors and capabilities, but it rarely provides any general solutions. Furthermore, the majority of soft robots developed so far do not contain sensors and thus operate in an open-loop mode. They fail to negotiate sensorimotor coordination, which is essential for the interaction of robots with their environment and to achieve autonomous behaviors. Thus one aim of this thesis is to investigate sensorimotor control for soft robots.

As increasingly recognized in recent years, both the scientific and the engineering fields can benefit from research collaborations in bio-inspired robotics [Beer et al., 1997] and neurorobotics [Edelman, 2007]. On the one hand, biology and neuroscience provide the knowledge and models on biological systems, from which robotics can take inspiration [Krieger et al., 2000; Pfeifer et al., 2007]. On the other hand, robotics research provides insight into the behaviors and neural control mechanisms of biological systems [Webb, 2000; Webb, 2002; Halloy et al., 2007], and can even be a powerful tool to generate predictions of living system [Ijspeert et al., 2007]. Researchers in the fields of octopus biology and soft robotics can also benefit from close collaboration with each other. We aim to study the octopus to extract key factors in controlling a completely soft body and

to develop a sensorimotor control scheme for octopus-inspired soft robots. In the next section, we present the reasons to learn from the octopus by summarizing interesting octopus characteristics and behaviors first; we then review the octopus sensorimotor control system and mechanism.

1.4 Learn from the Octopus

The octopus *vulgaris* is a marine invertebrate and belongs to the cephalopod mollusc. This animal has a completely soft body and at the same time exhibits sophisticated behaviors. It is considered to be the most intelligent and behaviorally flexible invertebrate. Therefore, studying the octopus has enormous potential for coming up with effective control strategies for a completely soft body without any rigid structure, as the octopus apparently has solved the control challenges.

1.4.1 Octopus Sensorimotor Control Capabilities

The octopus has remarkable abilities to control its soft body and eight flexible arms to perform various movements in the complex and unstructured underwater environment. For example, reaching toward a target [Gutfreund et al., 1996], precise point-to-point fetching [Sumbre et al., 2005; Sumbre et al., 2006], bipedal locomotion [Huffard et al., 2005; Huffard, 2006], and even defensive tool use [Finn et al., 2009]. Arms of the octopus can bend in any direction at any point along the arm and can elongate, shorten, and twist [Mather, 1998]. As an invertebrate, it has developed abilities and properties similar to vertebrates [Packard, 1972].

Because of its sensorimotor control capabilities, the octopus forms a valuable source of inspiration for soft robotics. Investigating and conceiving the underlying principles and control mechanisms that bring about the octopus sensorimotor capabilities would give insight of control strategies for soft robots.

1.4.2 Octopus Sensorimotor Control Strategy

The octopus has recourse to a rich sensory system and a distributed motor control scheme to achieve efficient sensorimotor control. We review and summarize related biological studies in this subsection.

Octopus Sensory System

The octopus is outstanding in its multi-modal sensation capabilities, which allow the animal to collect visual, mechanical, and chemical stimuli from the environment and to generate appropriate responses. The octopus' sensory system can be divided into two main categories.

The first category serves to probe the environment and mainly consists of the visual system and a large number of tactile and chemosensitive sensors. The octopus employs a sophisticated,

bilaterally symmetrical visual system, which consists of two eyes placed laterally on the head [Young, 1971]. The potential field of vision is 360 degrees, with a binocular vision of 5 degrees anterior and posterior of its head [Wells, 1978]. Therefore, the animal can visually guide its arms to reach or move toward a target in any direction [Gutnick et al., 2011]. Biological studies [Byrne et al., 2006b; Byrne et al., 2006a]) found that the octopus' arm choice is strongly influenced by visual information and the animal mostly uses the arm that is in a direct line between a target and the eye used. Tactile and chemosensitive cells are chiefly embedded in the octopus' suckers and arm skin to assist controlling the movements of individual arms [Wells, 1960; Wells, 1964; Young, 1965; Wells, 1978].

The second category is composed of a set of proprioceptive sensors monitoring muscle deformation. Proprioceptive sensors are embedded within the intrinsic muscles of the arms and involved in generation of arm movement [Graziadei, 1971; Wells, 1978]. Recent studies with isolated octopus arm nerve cords have demonstrated physiologically that the afferent proprioceptive information fed back into the arm's nervous system is seemingly important for controlling arm extension [Gutfreund et al., 2006].

Sensory information is also found to be locally processed. Based on morphological data [Graziadei, 1971] and physiological experiments [Rowell, 1963; Rowell, 1966], it is suggested that the signals from the octopus arm's sensory systems are processed at the level of the elaborated peripheral nervous system (PNS). The processed sensory information is transmitted to the central brain via a relatively small number of relay neurons in the axonal tract of the arm nerve cord [Young, 1971]. The local processing of sensory information suggests a distributed octopus motor control system.

Octopus Motor Control Mechanism

The octopus simplifies the control complexity of its completely soft body by using global variables generated by the central nervous system (CNS) and locally embedded motor programs in the PNS [Flash and Hochner, 2005].

The octopus CNS has been found to use a control strategy based on restricting control parameters to just three global variables for two vastly different arm movements: reaching and fetching. Figure 1.5 shows the sequences of the octopus reaching and fetching movements. The reaching movement is performed in a stereotyped manner [Gutfreund et al., 1996; Yekutieli et al., 2007]. It was discovered that the reaching movement begins with the formation of a bend somewhere along the arm. The bend then propagates toward the tip of the arm and the bend propagation depicts an invariant tangential velocity profile. Because the motion is restricted to a two-dimensional (2D) plane, the reaching movement is constrained to only three DoFs: two for the direction and one for scaling the bend propagation speed. A similar simplification strategy has evolved for the generation of the fetching movement [Sumbre et al., 2005]. The octopus uses the fetching movement to accurately bring an object to its mouth by creating an arm-like articulated

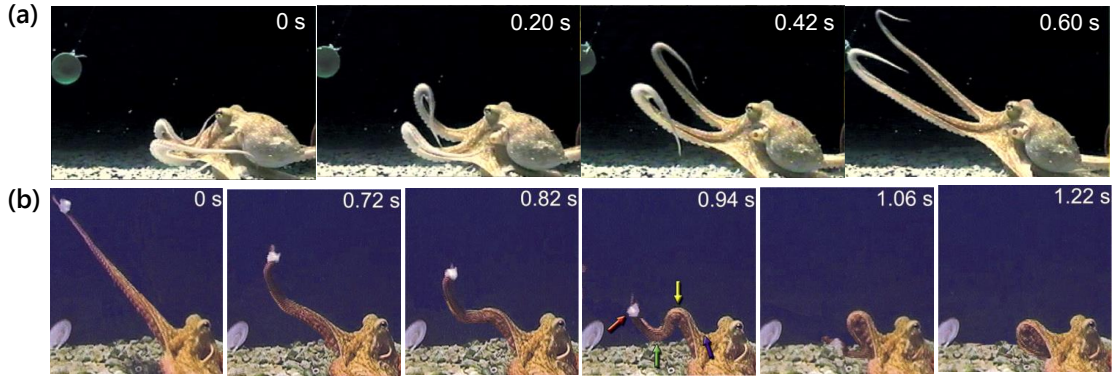


Figure 1.5: Sequences of octopus reaching and fetching movements (adapted from [Yekutieli et al., 2007; Sumbre et al., 2006]). (a) The reaching movement is featured by a bend propagation along the arm from the base to the tip; (b) the fetching movement is characterized by the formation and rotation of an arm-like articulated structure.

structure. The articulate structure is composed of three bends which behave like joints in skeletal structures. The proximal and the medial segments of the arm have nearly identical lengths. Unique to the octopus, however, this articulated structure is adjusted for each fetching movement according to the location along the arm where the object was grasped. Following the formation of the articulated structure, the distal bend is accurately moved towards the base of the arm by rotating the distal segment around the medial joint to bring the object to the month. Kinematic analysis shows that, again, the movement is restricted to a 2D plane and involves only three DoFs: one for each joint.

Many of the complex octopus arm movements are organized at the level of the PNS [Altman, 1971; Wells, 1978]. Combining behavioral and physiological studies, biologists revealed that the control complexity of the visually guided reaching movement is reduced by the existence of locally embedded motor programs for generating basic motion patterns, and that these programs are embedded within the arm's neuromuscular system [Sumbre et al., 2001]. This claim has been further supported by the phenomenon that electrical or tactile stimulations evoke extension in arms whose connection with the brain has been cut, and that these extensions showed identical kinematics to those of a natural octopus arm. This study suggested that brain commands are issued only for scaling, adjusting, and combining movements to achieve the desired result. In addition, the study of dynamic control of fetching movements reveals an extremely interesting mechanism for a simple calculation of the segment lengths according to where the object is grasped along the arm [Sumbre et al., 2005]. Two waves of muscle activation were detected: one started near the base of the arm and the other was close to the grasping site. These waves traveled toward each other and their collision point set the location of the medial joint. This mechanism allows calculating the structure of the articulated arm at the peripheral level without the need for complex representation of the eight arms in the relatively small octopus brain [Sumbre et al., 2006].

A recent study shows that sensorimotor integration is crucial for the determination of the animal's arm states and consequently for the generation of complex behavior, because the octopus CNS does not have a somatotopic organization of the motor areas [Zullo et al., 2009].

It is true that biologists have conducted a fair amount of studies on octopus behaviors and neurophysiology and these results are critical to soft robotics research. However, the investigation of the octopus sensorimotor coordination has been focused on a limited set of behaviors, specifically reaching and fetching. Building an autonomous robot, however, requires more complete information [Webb, 2000], so we need additional methodologies from the robotics field to compensate the limited biological studies.

1.5 Research Methodologies

With inspiration from studies on octopus sensorimotor control mechanisms, we probe the challenges of controlling soft robots by employing a primarily synthetic approach, distributed control, dynamical systems, and morphological computation, described in this section.

1.5.1 Synthetic Approach

Colloquially known as "understanding by building", the synthetic approach in robotics serves to provide insights into the behavior and sensorimotor control mechanism of living creatures by building robotic models. Previous studies [Pfeifer and Scheier, 1999; Pfeifer and Bongard, 2006] have discussed in depth the theories and practices of this bottom-up approach. There is a fundamental difference between the synthetic approach and the analytical approach used in, for example, neuroscience. In neuroscience, scientists routinely try to understand the function of a brain area by removing specific parts from the studied animal. Although much has been learned in this top-down approach, often the interesting interaction among different body parts is also eliminated. We aim to investigate the sensorimotor control of soft robots and also provide speculation on the octopus' neural control mechanism through the synthetic approach.

1.5.2 Distributed Control

Centralized control schemes typically rely on the exact kinematic and dynamic models of robotic platforms, for example, set-point and tracking controllers [Mochiyama and Suzuki, 2003]. However, it is difficult to get the exact analytical models for a soft robot. Distributed control provides an alternative approach for this scenario. The control functionalities of the octopus, as well as the control scheme used in this study, are distributed into three levels: the mechanism structure, the central level, and the peripheral levels, as shown in Figure 1.6.

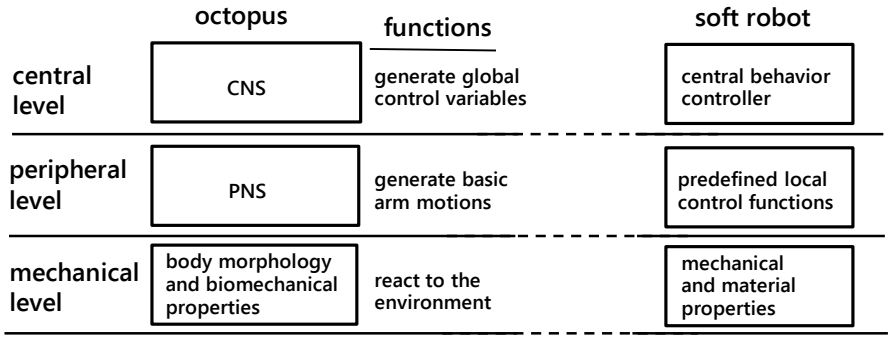


Figure 1.6: Three levels of distributed control. The overall behavior of the octopus and soft robots is governed by the interaction of the three levels.

Mechanical Intelligence

The principle of embodiment, the concept that intelligence requires a body, has shifted robotics research away from the traditional view which attributed the generation of adaptive behavior to control and computation. Instead, this approach is fundamentally based on the observation in nature that adaptive behavior emerges from the complex and dynamic interaction between the robot's morphology, sensorimotor control, and the environment [Pfeifer and Scheier, 1999; Pfeifer and Bongard, 2006]. This principle has been adopted in a wide range of approaches to the development of intelligent artifacts [Pfeifer et al., 2007]. The octopus reduces control complexity and improves movement efficiency by exploiting, or actively adjusting, its body properties. Living in the underwater environment, the octopus is almost free from the influence of gravity because its body density is just slightly heavier than that of the water; however, its movement speed is constrained by water drag forces [Mather, 1998]. Experiments show that the perpendicular drag coefficient is nearly 50 times larger than the tangential direction for a moving octopus arm [Yekutieli et al., 2005a]. To minimize the drag forces and thus achieve higher movement efficiency, only a small portion of the arm faces perpendicularly to the moving direction by actively adjusting the arm stiffness. During the fast escaping movement using a water jet, the distal parts of the soft octopus arms are naturally brought together by the water drag forces. This passive change in body configuration can significantly reduce the overall resistance from the water and increase velocity. In this study, based on the investigation of the anatomy of the octopus arm, the mechanical intelligence is considered for developing the first level of control embedded in the structure and the material properties.

Peripheral Sensorimotor Control

There is evidence for a highly distributed motor control in the octopus, as discussed in Section 1.4. The majority of the control for the arm is located in the arm itself [Sumbre et al., 2001; Yekutieli

et al., 2005a]. In accordance with this distributed control strategy, the octopus' CNS is relatively small compared with its PNS. Basic motion patterns have been identified in the octopus PNS [Gutfreund et al., 1996; Gutfreund, 1998; Yekutieli et al., 2002; Flash and Hochner, 2005]. The complexity of the visually-guided arm extension has been found to be reduced by hard-wired programs for the execution of the basic motion patterns [Sumbre et al., 2001]. The control scheme developed in this study for achieving efficient sensorimotor control of octopus-inspired soft robots is based on similar strategies for the reduction of DoFs and on hard-wired motor programs.

Central Behavioral Control

In the octopus, commands from the brain seem to be issued only for scaling, adjusting, and combining the basic arm movements to achieve the desired results [Sumbre et al., 2006]. The sensorimotor control for octopus-inspired soft robots developed in this study is based on the investigation of the parameters for coordinating the locally embedded motor programs. For example, the developed multi-arm soft robot coordinates the movements of all its arms in locomotion: each arm is controlled by the peripheral sensorimotor control, but the central behavioral control sends coordination commands for direction and speed.

Moreover, we need a theoretical and engineering tool to implement and analyze the distributed control approach described in the previous sections.

1.5.3 Dynamical Systems and Implementation in Recurrent Neural Networks

The dynamical systems approach enables us to take both the robot body dynamics and control system into account to generate a coherent control structure with appropriate functionalities in a bottom-up manner. The coupling between its body dynamics and the control system is of key importance in an embodied robot. If we predefine the control system in a top-down manner, we often tend to miss the role played by the body structure and material properties. In addition, the dynamical systems approach is also biologically plausible in view of anatomical [Gray, 1970] and physiological [Shomrat et al., 2011] studies on the octopus vertical lobe system. These studies revealed the existence of recurrence in octopus CNS, illustrated in Figure 1.7. Therefore, the dynamical systems approach is not only a promising mean for emulating the octopus CNS structure and functionality, but is also relevant due to the complexity of the soft robot body dynamics.

Recurrent neural networks (RNNs) are powerful tools to implement dynamical systems. They are usually governed by a huge number of control parameters such as the weights of connections, the bias, and the threshold function for each neuron, and thus are capable of approximating any function. They also have an ability to encode context-dependent control, that is, control dependent on the history of the RNNs' states. When an RNN is used as a control system, it is formally

described as a dynamical system with external perturbations (input). It has special applicability to sequential control tasks, which are expected to be a large constituent of the octopus' behaviors.

Learn the Soft Body Dynamics

Soft-bodied robotic systems have complex nonlinear intrinsic body dynamics. It is often infeasible to obtain analytical models of these robots and adopt traditional model-based control methods. Learning approach provide an alternative to enable a soft robot controller incorporating the complex robot body dynamics into the controller parameters and having a black-box model. In this study, this approach means to embed soft robotic body dynamics into the parameters of RNNs by applying proper learning algorithms.

To adapt the RNN parameters, supervised, unsupervised, and reinforcement learning schemes are available [Jaeger, 2002; Haykin, 2008]. The selection of a particular learning scheme depends on a number of factors, for example training time, application area, and availability of training data. In the context of this study, training data in the form of empirically observed natural octopus arm and body movement trajectories are available for various behaviors, and thus a supervised learning scheme is suitable.

Supervised learning of the RNN traditionally uses gradient-descent-based techniques to update connection weights and minimize the total error. To compute the error gradient, backpropagation through time or real-time recurrent learning is commonly used [Werbos, 1988; Williams and Zipser, 1989]. Gradient-descent-based techniques and several other more recent learning algorithm, such as extended Kalman filter, generally suffer from the following limitations [Jaeger, 2002; Hammer et al., 2009; Lukosevicius and Jaeger, 2009]:

- Long training time. Two factors result in the long training time: one is that each update

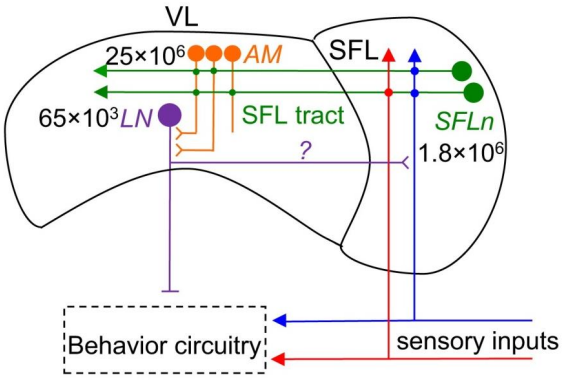


Figure 1.7: Sensory input recurrence in the octopus vertical lobe (VL) and superior frontal lobes (SFL). SFL neurons (SFLn) project to the VL via the SFL tract and innervate with the amacrine interneurons (AM). These AM then converge to large efferent neurons (LN). Sensory input reach behavior circuitry via two pathways and exhibit recurrence. *Note: the question mark means the indicated pathway is not sure to exist; this figure is adopted from [Shomrat et al., 2011].*

cycle during the learning can be computationally expensive; the other is that learning rate is normally chosen to be small to avoid instability and thus thousands of cycles are required because of slow convergence.

- Suboptimal solutions. Backpropagation methods are known to find only a local minimum. Meanwhile the gradual update of network parameters may drive the network dynamics through bifurcation, thus convergence is not guaranteed.

- Difficult to apply. These learning algorithms generally require advanced mathematical treatment and specially designed network topologies to achieve each specific task, thus plenty of experience is demanded.

- Poor extensibility. Because normally all the connection weights of an RNN are updated during learning, previously learned skill representations can be impaired or totally destroyed by subsequent learning.

As a result, these learning algorithms have not been widely utilized in practical engineering applications.

Reservoir computing (RC) approach provides a new paradigm for the design and supervised learning of RNNs [Jaeger, 2001; Maass et al., 2002; Steil, 2004]. In this approach, an RNN with random fixed internal connection weights is generated as the dynamical reservoir (DR) and only the connection weights from the DR to the readout units are adapted in learning [Verstraeten et al., 2007; Lukosevicius and Jaeger, 2009]. Essentially, any type of regressors can be used to generate the reservoir output from the DR, while simple linear regression algorithms are usually used in practice. This paradigm makes it possible to use any of the fast linear regression algorithms [Hastie et al., 2009] and offers some advantages over traditional methods, such as the ease of training, fast learning speed, and guaranteed optimality in a least square sense. Furthermore, the RC approach provides a simple solution to extend the functionality of RNNs — new skills are added by appending extra output units. Because the output weights of different output units are independent, previously learned skills are not impaired. There are mainly three approaches in RC: echo state network (ESN) [Jaeger, 2002; Jaeger and Haas, 2004], liquid state machine (LSM) [Maass et al., 2002], and backpropagation-decorrelation learning rules [Steil, 2004]. All of them have the potential to achieve the tasks investigated in this thesis, we adopted and explored the ESN and the LSM approaches.

1.5.4 Morphological Computation

Advances in the filed of bio-inspired robotics suggest that the specific body morphology and construction materials work in parallel with sensorimotor coordination in shaping the way an agent behaves [Pfeifer and Bongard, 2006]. A proper design of morphology and selection of materials can significantly reduce computational complexity by inducing statistical regularity in

sensory input and in the control target. Exploiting morphology to take over part of the control tasks is termed morphological computation, which has been explicitly or implicity employed by robot designers [Paul, 2006]. A number of recent theoretical studies [Hauser et al., 2011; Hauser et al., 2012] on morphological computation demonstrated the computation power of physical bodies. Moreover, these theoretical studies suggest that a computationally powerful physical body should have nonlinearity, compliance, and high dimensions in its state space, which are exactly some characteristics of a soft body. We explore morphology design to reduce control complexity of octopus-inspired soft robots and the feasibility to use the rich nonlinear body dynamics of soft robots as a computational device.

1.6 The OCTOPUS Integrating Project

The work described in this thesis was performed as a part of a large-scale EU-funded project, the OCTOPUS integrating project [1]. This project brought together roboticists, engineers, mathematicians, biologists, and neuroscientists from six research groups to investigate and understand the principles that give rise to the octopus sensorimotor control abilities by building octopus-inspired soft robots inspired by the morphology and behavior of the octopus. The aims were not only to inspire the development of new soft actuation systems, sensors, modeling and control techniques for a new generation of soft robots, but also to contribute to the scientific research into fundamental biological issues. The originality of the OCTOPUS project lies in the fact that it is the first comprehensive investigation into soft robotics.

1.7 Research Questions and Hypotheses

Employing the research methodologies and theoretical tools outlined in the previous sections, this thesis aims to identify and evaluate the key factors to achieve efficient sensorimotor control of octopus-inspired soft robots. Five specific research questions and research hypotheses are derived from this main goal.

There is a large volume of biological studies concerning the anatomy, biomechanics, neurophysiology, and behavior of the octopus. The biological inspiration from the octopus will be investigated as the first step; therefore, our first research question is:

Q1: What are the key factors in achieving the sensorimotor control abilities of the octopus?

Collaborating closely with biologists, we interpret the existing biological studies of the octopus with the methodologies outlined in the previous section. Our hypothesis is that the octopus biological study, combining with engineering tools from the robotics field, would provide a sensorimotor control scheme for soft robots.

[1] Project website: http://www.octopusproject.eu/

Then we move to implement the proposed sensorimotor control scheme and to evaluate it by reproducing typical octopus behaviors. We start with the reaching movement, which has been extensively documented in the biological literature. Our second research question is thus:

Q2: Is it possible to generate the stereotypical octopus reaching movement with the proposed sensorimotor control scheme?

Our hypothesis is that some additional stimulating factors, besides biological inspirations, in controlling a completely soft body would be revealed during the implementation and evaluation process.

Furthermore, octopus-inspired soft robots need to implement multiple behaviors from the octopus and adaptively select a behavior according to the interaction with the environment. The third question is therefore:

Q3: How can multiple behaviors be embedded and switched in the sensorimotor control scheme?

The study of the octopus neural system revealed the non-somatotopic organization of the higher motor center [Zullo et al., 2009]. This study suggests that the circuits control the octopus behaviors overlap in the same region of its brain, rather than in multiple distinct regions. The octopus seems to embed a large number of behaviors in a "behavior reservoir". The RC paradigm [Jaeger, 2002; Jaeger and Haas, 2004] provide a possible scheme of the way that the octopus manages to embed multiple behaviors and switch among them. Our hypothesis is thus that the RC approach is a feasible way to embed and switch among multiple behaviors in octopus-inspired soft robots.

The octopus has eight almost identical arms, but the multi-arm coordination mechanism has been rarely studied. Next, we investigate the multi-arm coordination mechanism and try to answer the question:

Q4: How can multiple arms be coordinated to achieve goal-oriented movement in a multi-arm soft robot?

Specifically, we evaluate if the same control scheme and implementation used for a single soft arm can be easily adapted to achieve the multi-arm coordination task. Our hypothesis is that we can use the same control implementation to achieve this task, and some insights about the octopus multi-arm coordination mechanism would be generated.

Finally, one significant characteristic of soft robots is their completely soft bodies, which exhibit complex nonlinear dynamics. Although it is a challenge to control a completely soft body using traditional engineering tools, the complex body dynamics could potentially be a useful resource. We ask the question:

Q5: Is it possible to exploit soft body dynamics as a computational device?

Because the DR in the RC approach is just a randomly generated RNN, we hypothesize that we can sense and exploit the diverse soft body dynamics as the physical reservoir.

1.8 Overview

This thesis is organized around six papers, which have been published in, or submitted to, peer-reviewed scientific journals or conferences. Figure 1.8 shows an overview of the thesis. The remaining chapters are partitioned into three main topics: an octopus-inspired soft robot control scheme (Chapter 2); implementation and evaluation of the control scheme on simulated and physical soft robots inspired by the octopus morphology (Chapters 3 – 6); exploiting the rich body dynamics of soft robots as a computational resource (Chapter 7).

Chapter 2 presents a sensorimotor control scheme for octopus-inspired soft robots through extensive review and interpretation of biological studies on the animal. The proposed control scheme serves as the basis and starting point of further investigation.

In Chapter 3, we implement the control scheme by ANNs and train the ANNs to achieve a goal-oriented reaching task. A 2D octopus arm simulator is used to evaluate the controller.

Chapter 4 studies how to embed multiple behaviors and switch among them. We examine the feasibility of our approach on a physical soft robotics arm equipped with different types of

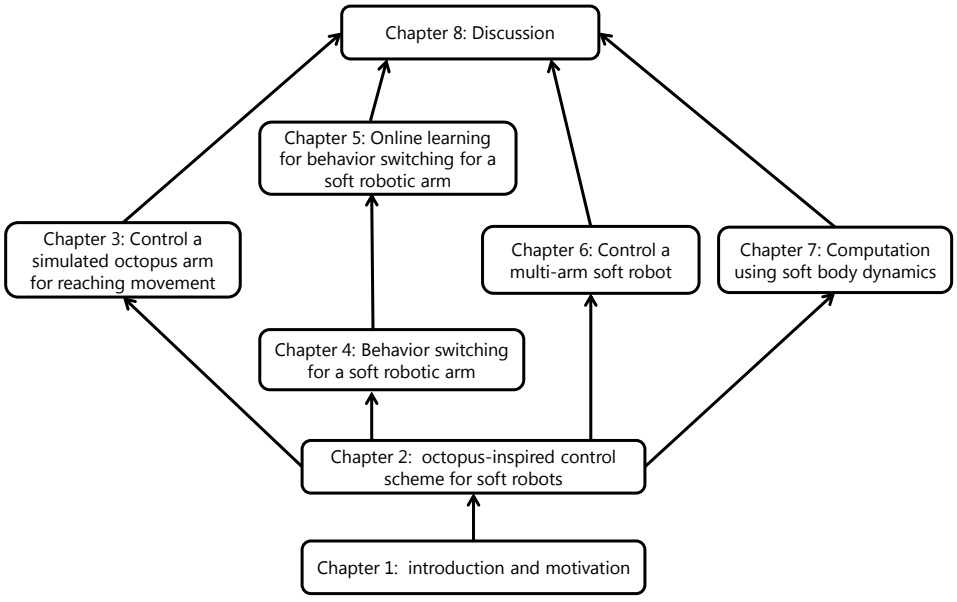


Figure 1.8: Thesis overview. Based on the review in Chapter 1, we present a octopus-inspired control scheme for soft robots in Chapter 2. The control scheme is implemented and evaluated on a single simulated octopus arm (Chapter 3), a single physical soft robotic arm (Chapters 4 – 5), and a multi-arm physical soft robot (Chapter 6). Chapter 7 presents a study to exploite physical soft body dynamics to do computation. Chapter 8 summaries the studies covered in the thesis and discusses the results.

sensors.

Chapter 5 serves as an improvement of the study presented in Chapter 4. It adopts an online learning method to accomplish the same task on the same platform.

Chapter 6 studies the sensorimotor control of a multi-arm soft robot. The robot is trained to autonomously approach an object.

Chapter 7 investigates the potential of sensing and using the soft robot body dynamics as a computational device. We embed several flexible bend sensors inside the soft robot and treat each sensor as a node of an ANN. The capacity of the physical neural network is quantified by memory capacity and closed-loop control tasks.

Finally, Chapter 8 concludes this thesis by discussing its implications for both octopus biology and soft robots, as well as recommendations for future work.

Chapter 2

Octopus-Inspired Sensorimotor Control Architecture for Soft Robots

This chapter is a summary of our publication [Li et al., 2011], which is enclosed as Appendix A.

Li, T., Nakajima, K., Kuba, M., Gutnick, T., Hochner, B., and Pfeifer, R. (2011). From the octopus to soft robots control: an octopus inspired behavior control architecture for soft robots, *Vie et Milieu/ Life and Environment*, 61(4): pp. 211–217.

In what follows, we present the abstract and summarize the results related to the main topic discussed in this thesis.

Abstract: *In recent years, breakthroughs have been made in both biology and robotics by the close cooperation between biologists and roboticists. Researchers in the fields of octopus biological study and soft robotics can also benefit from working together and inspiring each other. However, this collaboration is not easy because of the different research motivations and terminologies used. This paper starts from challenges for controlling soft robots, through biological inspirations from the octopus, to their engineering interpretation for creating a behavior control architecture for soft robots. Hypotheses related to the octopus biology from the engineering perspective are also proposed. This work serves as a case study in biologist-roboticist collaboration. It seeks to promote mutual understanding and cooperation between these two disciplines.*

2.1 Results

By summarizing and interpreting the biological studies on octopus anatomy, behavior, neurophysiology, and following the framework of embodiment, we proposed a novel octopus-inspired sensorimotor control architecture for soft robots. This study addressed the following points in

controlling octopus-inspired soft robots:

- **Control unlimited DoFs by distribution of functionalities.** A soft body contains virtually unlimited number of DoFs, which contributes to the operation flexibility. Meanwhile, it is a great challenge to efficiently control these DoFs. Inspired by the octopus biological studies [Sumbre et al., 2001; Sumbre et al., 2005], we followed a distributed control strategy to reduce the control complexity of the large number of DoFs. The high-level behavior controller only initiated the motion, while the highly autonomous low-level movement controller assumed a large part of control functions.

- **Harness soft body dynamics by proper timing.** Biological studies show that the octopus CNS uses a limited parameters to specify its reaching movement [Yekutieli et al., 2005a; Yekutieli et al., 2005b]. The slow body dynamics of a soft body results in a time delay to transfer the applied forces from the proximal to the distal part. We proposed that the key factor in harnessing a soft arm is to control all the muscles together by initiating the activation of the embedded motor programs in the peripheral level with appropriate timing instead of manipulating individual muscles.

- **Learn soft body dynamics.** A soft robot central behavior controller needs knowledge of the specific soft body dynamics to apply control commands at proper timing. However, it is in general difficult to build a precise mathematical model and implement model-based control for soft robots. An alternative method is to learn an implicit model of the soft body and embed the specific body dynamics into the controller parameters. We adopted the dynamical systems approach to learn and embed the implicit model of a soft body.

2.2 Contributions

This study investigated some of the major control challenges for octopus-inspired soft robots. The main contributions are as follows:

- Synthesized a novel sensorimotor control architecture to tackle the intrinsic challenges in controlling soft robots.

- Proposed some biological hypotheses within the framework of embodied intelligence and from the engineering point of view.

- By showing our methodologies and how such research might benefit both engineering and biological fields, it would promote mutual understanding and cooperation between biologists and roboticists.

Chapter 3

Control a Simulated Octopus Arm for Reaching Movement

This chapter is a summary of our publication [Nakajima et al., 2011b], which is enclosed as Appendix B.

Nakajima, K., Li, T., Kuppuswamy, N., and Pfeifer, R. (2011). Harnessing the Dynamics of a Soft Body with "Timing": Octopus-Inspired Control via Recurrent Neural Networks, In *the 15th International Conference on Advanced Robotics*, pp. 277–284.

In what follows, we present the abstract and summarize the results related to the main topic discussed in this thesis.

Abstract: *This study aims to explore a control architecture that enables the control of a soft and flexible octopus-like arm for an object reaching task. Inspired by the division of functionality between the central and peripheral nervous systems of a real octopus, we discuss that the important factor of the control is not to regulate the arm muscles one by one, but rather to control them globally with appropriate timing, and we propose an architecture equipped with a recurrent neural network (RNN). By setting the task environment for the reaching behavior, and training the network with an incremental learning strategy, we evaluate whether the network is then able to achieve the reaching behavior. As a result, we show that the RNN can successfully achieve the reaching behavior, exploiting the physical dynamics of the arm due to the timing based control.*

3.1 Results

In this study, we investigated the challenge to generate the stereotypical octopus reaching movement on a single simulated octopus arm. This study addressed the following specific points:

- **Control mechanism of the octopus reaching movement.** The octopus stereotypical reaching movement is one of the most studied behaviors by biologists [Gutfreund et al., 1996; Gutfreund, 1998; Yekutieli et al., 2005a; Yekutieli et al., 2005b; Gutfreund et al., 2006; Yekutieli et al., 2007]. The animal has evolved simplification strategies to reduce the control complexity of its completely soft arms. That is, the division of functionality between its CNS and PNS: the CNS only sends initiating and modulating commands to the PNS; the PNS directly handles all the muscles in the soft arms. In addition to biological inspirations, we examined the coordination between the CNS and the PNS: the way that the CNS initiated a sequence of commands at a proper time. As a result, timing was suggested as an additional important factor in control a soft arm.

- **Implementation of the control mechanism.** Motivated by this division of functionality between the octopus CNS and PNS, we implemented the control mechanism by a RNN and a feed forward network (FFN). The RNN mimicked the octopus CNS and generated initiation command sequences at a proper time; while the FFN emulated the octopus PNS and controlled arm muscles at each control time step.

- **Evaluate the control mechanism on a physical simulator of the octopus arm.** We developed a physical simulator of an octopus arm to evaluate the control mechanism. The advantages of a simulator lie in the fact that almost all the parameters can be precisely set and examined and that long term experiments are possible thanks to its relatively better stability compared with a physical structure. By setting the task environment for the reaching behavior and training the network with an incremental learning strategy, the network was able to achieve the reaching behavior. Finally, we analyzed how the combination of the RNN and the FFN successfully achieves the reaching behavior using dynamical systems theory.

3.2 Contributions

This study was the first step on implementing and evaluating the octopus-inspired control architecture: achieving a single behavior on a single simulated soft octopus arm. The main contributions of this work are as follows:

- Implemented and evaluated the control scheme proposed in Chapter 2 on a simulated octopus arm for a reaching task using a RNN and a FFN.

- Demonstrated that the important factor in controlling a soft octopus arm is not to regulate the arm muscles one by one, but rather to control them globally with appropriate timing.

- Analyzed how the reaching behavior was achieved by examining the RNN recurrent dynamics.

Chapter 4

Behavior Switching Using Reservoir Computing for a Soft Robotic Arm

This chapter is a summary of our publication [Li et al., 2012b], which is enclosed as Appendix C.

Li, T., Nakajima, K., Cianchetti, M., Laschi, C., and Pfeifer, R. (2012). Behavior Switching by Using Reservoir Computing for a Soft Robotic Arm, In *the 2012 IEEE International Conference on Robotics and Automation*, pp. 4918–4924.

In what follows, we present the abstract and summarize the results related to the main topic discussed in this thesis.

Abstract: *Soft robots have significant advantages over traditional robots, which are commonly made with rigid materials. However, controlling this type of robot by conventional approaches is difficult. Reservoir computing has been demonstrated to be an effective approach for achieving rapid learning in benchmark tasks and conventional robots. In this study, we investigated the feasibility and capacity of the reservoir computing approach to embedding and switching among multiple behaviors in online manner in a soft robotic arm. The result shows that this approach can successfully achieve this task.*

4.1 Results

Our study on the simulated octopus arm in Chapter 3 has focused on only the reaching behavior. This study investigated the challenge to embed multiple behaviors in a controller and autonomously switch among these behaviors on a physical soft robotic platform and addressed the following specific points:

- **Learn and switch among multiple behaviors.** As any agent aims at achieving autonomy, a soft robot needs to equip with multiple behaviors and to autonomously switch among the behaviors during the dynamical interaction between the robot and its working environment. Traditional learning methods for the RNNs have poor extensibility and thus cannot easily incorporate multiple behaviors. We introduced the RC approach, which is an easy and fast way to train the RNNs, for this task.
- **Evaluate the reservoir computing approach on a physical soft robotic platform.** To further evaluate the control scheme proposed in Chapter 2, we developed a physical soft robotic platform to study the soft robot control challenges under real-world constrains. A physical robotic platform is prone to frequent failures compared with a simulator, so adopting fast and easy learning methods is essential. We evaluated the reservoir computing approach, specifically the ESN, on the physical soft robotics platform and demonstrated the feasibility to use this approach to achieve the multiple behavior learning and switching task.
- **Control robustness to sensor noises.** One important criterion in evaluating a controller for physical platforms is its robustness to noises. Different type of sensors exhibit distinct noise characteristics. We utilized position and force sensors in our experiments. Our results showed that although both types of sensors could achieve the desired task but a noisier pattern was observed when force sensors were adopted.

4.2 Contributions

We evaluated the control scheme on a physical soft robotic platform for multiple behaviors. The main contributions of this work are as follows:

- Constructed a soft robotic platform inspired by the morphology of an octopus arm and equipped with position and force sensors.
- Introduced a fast and stable way to construct and train the RNNs on a physical soft robotic platform.
- Demonstrated that the RC approach can be used to embed and switch among multiple sequential behaviors in a physical soft robotic platform.

Chapter 5

Online Learning Technique for Behavior Switching in a Soft Robotic Arm

This chapter is a summary of our publication [Li et al., 2013], which is enclosed as Appendix D.

Li, T., Nakajima, K., and Pfeifer, R. (2013). Online Learning Technique for Behavior Switching in a Soft Robotic Arm, In *the 2013 IEEE International Conference on Robotics and Automation*, pp. 1288–1294.

In what follows, we present the abstract and summarize the results related to the main topic discussed in this thesis.

Abstract: *Soft robots possess several potential advantages over traditional articulated ones and have attracted significant interest in recent years. However, to control this new type of robot using conventional model-based robotic control approaches is generally ineffective. In this paper, we investigate the challenge to embed and switch among multiple behaviors for an octopus-inspired soft robotic arm. An online learning method for reservoir computing is exploited for this task. This online learning method does not require a separate teaching data collection phase; thus, it has the potential to achieve autonomy in soft robots. Our result shows the feasibility of this approach.*

5.1 Results

The work presented in Chapter 4 has two limitations: (1) the control signals used to switch the embedded behaviors were arbitrarily defined numbers; (2) the learning of the ESN was offline and required a separate phase to collect training data. This study addressed these limitations and presented the following specific points:

- **Behavior switching according to visual input.** The octopus behavioral studies have suggested that visual information is one of the most essential exteroceptive sensory modalities to the animal and triggers behavior change [Byrne et al., 2006a; Byrne et al., 2006b]. The two octopus optic lobes consist of about 20 percents of the total neurons in the octopus nervous system [Wells, 1978]. Following the importance of the octopus visual modality, we improved our soft robotic system by incorporating a camera. Different objects were put in front of the camera and used as the signals to switch among different behaviors.
- **Online learning for the RC.** Online learning methods have the potential to adapt to unexpected changes in the robot structure and the environment, without taking the system offline. This advantage is critical to achieve the autonomy of robots. We demonstrated that an online learning method for the RC could be implemented on the same octopus-inspired soft robotic arm, presented in the previous chapter, to embed and switch among multiple behaviors by visual input. The performance of the RC was evaluated by comparing the desired outputs and the actual outputs. We also analyzed the quality of the learning by plotting the output errors during the learning and evaluation phases, as well as the trajectories of the amplitudes of output weights during learning.

5.2 Contributions

This study demonstrated the possibility to embed and switch among multiple behaviors using an online learning method and visual input. The main contributions of this work are as follows:

- Experimentally demonstrated the biological hypothesis that behavior changes in the octopus is triggered by visual input, on a physical soft robotic platform.
- Demonstrated the possibility of using an online learning method to embed and switch among multiple behaviors on a physical soft robot platform. The online learning method provided a possible step toward achieving autonomy for soft robots.

Chapter 6

Octopus-Inspired Sensorimotor Control of a Multi-Arm Soft Robot

This chapter is a summary of our publication [Li et al., 2012a], which is enclosed as Appendix E.

Li, T., Nakajima, K., Calisti, M., Laschi, C., and Pfeifer, R. (2012). Octopus-Inspired Sensorimotor Control of a Multi-Arm Soft Robot, In *2012 IEEE International Conference on Mechatronics and Automation*, pp. 948–955.

In what follows, we present the abstract and summarize the results related to the main topic discussed in this thesis.

Abstract: *Soft robots have significant advantages over traditional rigid robots because of their morphological flexibility. However, the use of conventional engineering approaches to control soft robots is difficult, especially to achieve autonomous behaviors. With its completely soft body, the octopus has a rich behavioral repertoire, so it is frequently used as a model in building and controlling soft robots. However, the sensorimotor control strategies in some interesting behaviors of the octopus, such as octopus crawling, remain largely unknown. In this study, we review related biological studies on octopus crawling behavior and propose its sensorimotor control strategy. The proposed strategy is implemented with an echo state network on an octopus-inspired multi-arm crawling robot. We also demonstrate the control strategy in the robot for autonomous direction and speed control. Finally, the implications of this study are discussed.*

6.1 Results

We have focused on the sensorimotor control of octopus-inspired soft robots at the single-arm level in the previous studies (Chapters 3–5). This study investigated the multi-arm coordination

issue and addressed the following points:

- **Sensorimotor control scheme of the octopus crawling behavior.** We implemented and extended the sensorimotor control strategy proposed in Chapter 2 to study the octopus crawling behavior, which is a representative task of multi-arm coordination. Crawling is a commonly observed but rarely studied octopus locomotion mode. Only a recent study briefly reported the observation of basic crawling movement sequence and arm patterns [Calisti et al., 2011]. By reviewing the octopus crawling behavior, sensory system, and nervous system, we summarized the information flow in the octopus crawling behavior, shown in Fig. 1 of Appendix E. The control of octopus crawling is distributed between the CNS, the PNS, and the octopus body properties.

- **Experimental evaluation of the multi-arm control scheme.** To evaluate the control scheme, we performed experiments on a physical multi-arm soft robotic platform to achieve autonomous direction and speed control. Biological studies on the octopus showed the importance of visual information on arm choices and coordination [Byrne et al., 2006a; Byrne et al., 2006b]. Therefore, raw image data from an omnidirectional camera placed on top of the robotic platform was used as sensory inputs to mimic the 360 degrees visual capacity of the octopus. The octopus CNS function was implemented by the ESN, which has been shown to control a single soft robotic arm effectively in Chapters 4. Preprogrammed motor control programs were used to mimic the highly autonomous octopus PNS. We trained the ESN using collected teaching data and successful achieved the desired sensorimotor tasks. Meanwhile, the specification of the robotic platform and the performance of the control strategy were analyzed to evaluate the generalization potential.

6.2 Contributions

We extended the study of sensorimotor control of a single soft robotic arm to the multi-arm coordination scenario. The main contributions of this work are as follows:

- Proposed a distributed sensorimotor control strategy for the octopus crawling behavior by synthesizing pieces of biological studies on the octopus behavior, sensory systems, and neurophysiology.

- Built an octopus-inspired multi-arm robot as an experimental platform to evaluate the control scheme and to demonstrate the octopus crawling behavior.

- Implemented and demonstrated the proposed control strategy on the octopus-inspired, multi-arm robotic platform.

Chapter 7

Computation Using Soft Body Dynamics

This chapter is a summary of our publication [Nakajima et al., 2013a], which is enclosed as Appendix F.

Nakajima, K., Li, T., Hauser, H., Iida, F., and Pfeifer, R. (2013). Short-Term Memory in a Silicone-Based Soft Robotic Arm, submitted to *Nature Communications*.

In what follows, we present the abstract and summarize the results related to the main topic discussed in this thesis.

Abstract: *Soft materials are not only highly deformable, but they also possess rich and diverse body dynamics. Here we demonstrate that such soft body dynamics can be employed to conduct certain types of computation. Using body dynamics generated from a soft silicone arm, we show that they can be exploited to emulate functions that require memory and to embed robust closed-loop control into the arm. Our results suggest that soft body dynamics have a short-term memory and can serve as a computational resource. This finding paves the way towards exploiting softness for control of a large class of systems and a novel understanding of the notion of computation.*

7.1 Results

This study focused on the physical body level of the distributed control mechanism for octopus-inspired soft robots. We addressed the following specific points:

- **Use the body dynamics of a soft silicone arm as the DR of an ESN.** Inspired by the fact that the DR in an ESN is a randomly generated RNN which brings the reservoir inputs to high dimension to increase separation properties, we adopted the diverse nonlinear body

dynamics exhibited by a soft silicone arm as a physical reservoir. To sense the soft body dynamics, we embedded ten bend sensors into a soft silicone arm. We conducted experiments on an experimental platform to show that the soft silicone arm has the property of the DR and thus can be used as a physical reservoir.

- **Memory capacity of a soft robotic arm.** Using the diverse soft body dynamics to construct a timer, we showed that the soft silicone arm posses memory of its past motor input history through its transient body dynamics. Furthermore, we quantitatively characterized the intrinsic memory capacity of the soft silicone arm by using two function-emulation tasks: short-term memory and N-bit parity check. We also demonstrated that the two tasks can be performed simultaneously (multi-tasking ability).

- **Closed-loop control of a physical body using morphology computation.** Embodied intelligence suggests that adaptive behaviors emerge as the result of the interaction between the control, the body, and the environment. We used a closed-loop control task to demonstrate the sensorimotor coupling and that the soft silicone arm, as a physical reservoir, could control its own body to sustain periodic motions by generating the next required motor command.

7.2 Contributions

We presented and demonstrated an original approach in morphological computation and soft robot control by exploiting a soft silicone arm as a computational device. The main contributions of this work are as follows:

- Examined the engineering challenge to reliably embed multiple flexible sensors inside a soft silicone arm to detect its diverse nonlinear body dynamics while maintaining its flexibility. The design can be applied in other soft robot designs.

- Investigated the dynamic properties of the soft silicone arm and demonstrated the potential to use its diverse nonlinear body dynamics as the DR of an ESN by multiple benchmark tasks.

- Quantitatively studied the intrinsic computational capacity, especially memory capacity, of a soft silicone arm.

Chapter 8

Discussion

In the preceding chapters, we have presented a synthetic approach to study the sensorimotor control of octopus-inspired soft robots from several aspects. In this chapter, we summarize the results and discuss their implications for the octopus biology, as well as for soft robot sensorimotor control in general.

This chapter is structured as follows: First, we summarize the main results from the individual publications presented in Chapters 2–7. Then, we draw general conclusions and implications for the octopus biology and sensorimotor control of soft robots. Finally, we discuss future research directions in the design and control of soft robots.

8.1 Summary of Results

This thesis aims to investigate the key factors that contribute to the sensorimotor control of the octopus and octopus-inspired soft robots. We identified five specific research questions in Chapter 1 that were addressed in this thesis:

- **Q1: What are the key factors in achieving the sensorimotor control abilities of the octopus?**
- **Q2: Is it possible to generate the stereotypical octopus reaching movement with the proposed sensorimotor control scheme?**
- **Q3: How can multiple behaviors be embedded and switched in the sensorimotor control scheme?**
- **Q4: How can multiple arms be coordinated to achieve goal-oriented movement in a multi-arm soft robot?**
- **Q5: Is it possible to exploit soft body dynamics as a computational device?**

Sensorimotor control of octopus-inspired soft robots poses enormous challenges to traditional robotic techniques and control methods from multiple aspects, as summarized in Chapter 1. We investigated the key factors in handling these challenges and provided a framework to implement sensorimotor control for octopus-inspired soft robots (**Q1**) in Chapter 2. Combining biological inspiration with engineering knowledge, we proposed a distributed control scheme, in which the CNS (high-level control) only initiates motion by generating global parameters, while the PNS (low-level control) assumes a large part of control functions and is largely autonomous. The required control parameters for the unlimited DoFs of a soft body are reduced to a limited number of global parameters by this distributed control scheme. Take the octopus reaching movement as an example: the global parameters generated by the CNS are moving directions and speed; the locally embedded motor programs in charge of controlling each muscle and generating the basic bend propagation movement pattern. We included *timing* as another key factor to reflect the fact that a soft body is characterized by relatively slow body dynamics and that it is difficult to directly use sensors to reliably estimate the state of a soft body. The slow body dynamics is readily observed by the phenomenon that actuation forces and external disturbances need longer time to travel in a soft body than in a rigid one. The *timing* factor is especially critical when issuing a sequence of control actuation. The control sequence needs to be issued with a proper interval based on the specific soft body dynamics. For instance, the octopus reaching movement roughly consists of arm bend formation and propagation. The octopus CNS decides the proper time to initiate the bend propagation, that is, the bend formation should be finished first. Because the approaches to directly model soft structures have not been well developed, we included learning in our control scheme as a manner to embed soft body dynamics in the controller implementation. As a result, the controller should be able to issue a control sequence with a proper time interval based on the learned soft body dynamics. We also considered the sensor modalities required to close the sensorimotor loop. Combining all these factors, we finally synthesized a sensorimotor control scheme ready to be implemented and evaluated.

We implemented the control scheme and evaluated it by generating the stereotypical octopus reaching movement toward a target (**Q2**) in Chapter 3. The evaluation was conducted on a 2D octopus arm simulator, which was modeled after the muscular-hydrostatic properties of the octopus arm. The merit of a simulator is that it can get rid of engineering constraints in constructing soft robots; therefore, we could mimic the biomechanics of the octopus arm in detail. Moreover, a simulator enables us to explore the octopus sensorimotor control with more freedom because each parameter representing muscles can be precisely controlled and monitored. We implemented the control scheme by an arm activation function and two ANNs: a FFN and an RNN. The former mimics locally embedded motor programs in the octopus PNS, while the latter resembles the function of the octopus CNS. Specifically, the FFN provides a mapping from a target position to three global control parameters: arm activation strength and the two components of reaching direction in 2D. Meanwhile, the RNN mainly takes partial information about the arm state and decides the proper time to initiate the motor programs embedded in the FFN. Because

the proper time is determined by the soft arm dynamics, we used a supervised learning method to learn the dynamics and embed the dynamics into the parameters of the RNN. The teaching data was generated by randomly activating the soft arm and collecting the data in cases that it successfully reached the target. The implemented controller could adaptively generate reaching movements toward a target that appeared at a random position within reaching distance of the arm. Therefore, the sensorimotor control scheme proposed in Chapter 2 catches the essence to control a single soft arm.

After examining the generation of a single behavior, we continued to investigate embedding multiple behaviors and autonomously switching among them using the sensorimotor control scheme (**Q3**). A single-arm physical soft robotic platform was developed for this study. Compared with a simulator, a physical platform includes the actual complex interaction between the nonlinear soft body dynamics, the control input, and the environment without any subjective simplification. In Chapter 4, we implemented the sensorimotor control scheme with an easy and fast learning method because of the relative instability of a physical platform compared with a simulator. The high-level global parameter generator (the CNS) was implemented by an ESN; the locally embedded motor programs were achieved by the functions of motor control boards. The motor target positions and directions for the next timestep were used as the global control parameters. The implemented controller was shown to be able to learn the complex soft body dynamics and embed three oscillatory movements with supervised learning. These three movements could be switched autonomously according to an external switching signal. We also demonstrated that a FFN, however, could not achieve the same task. The failure of the FFN was because that the memory capacity of the ESN was needed to determine the proper time to generate the motor commands. In Chapter 5, we presented further improvement in two aspects: visual input from a camera replaced the arbitrary behavior switching signal; an online learning method, which did not need a separate teaching data collection phase, was adopted. This improved implementation satisfyingly achieved the same task. The online learning method provides a feasible way to achieve an autonomous soft robot which can compensate for the dynamics change from the body and the environment by online learning.

In Chapters 3–5, we investigated the sensorimotor control of a single soft arm. In Chapter 6, we extended our focus and studied the sensorimotor control mechanism concerning the coordination of multiple soft arms (**Q4**). We evaluated whether the same implementation used in Chapter 4 can be used for the multi-arm coordination scenario. The task used in this study originated from the biological study that the octopus arm usage is determined by the visual information. We implemented the visual modality with an omnidirectional camera, which mimicked the 360 degrees potential visual field of the octopus. After applying supervised learning, the same ESN-based control implementation could coordinate multiply soft arms in our robotic platform to achieve autonomous target tracking and speed changing according to the distance to the target. We were aware that the task used in the multi-arm coordination study may not require an ESN, as a FFN probably can achieve the same task. However, our aim in this study was not to test the potential

of the control implementation, but to show its extensibility for multi-arm coordination tasks. Up to this point, we demonstrated the capacity of our sensorimotor control scheme to control a single arm for a single sequence movement, to achieve multiple behavior embedding and autonomous switching, and to achieve multi-arm coordination.

To investigate the potential of using a soft robot body as a computational device (**Q5**), we built a soft silicone arm with several sensors embedded and conducted experiments in Chapter 7. It was not a trivial task to embed sensors inside a soft arm because of multiple technical constraints. First, there is only a limited choice of sensors that meet the requirements that we had. They should be flexible themselves, reliable enough to sustain long-term experiments, able to be embedded into silicone without degrading and losing functionalities, available in large quantity, and the sensor signals should be easily collected without the need for complicated data acquisition devices. Second, there is a trade-off between the number of sensors embedded inside the soft silicone arm and the requirement to maintain the arm softness. Third, there is a trade-off between the strength of electrical cables for the sensors and the requirement to keep the softness of the arm. Finally, the position and orientation of the sensors should reflect the spatiotemporal distribution of the diverse dynamics of the soft arm. We created a design that meets all these requirements through incremental iterations of development. The design is transferable to other soft robots design in general. We conducted experiments to show that the diverse body dynamics of the soft silicone arm can be used as a physical DR to achieve several benchmark tasks that requires memory and nonlinearity. Then, we demonstrated that the soft arm, as a physical reservoir, has enough computation capacity to generate periodic motion by predicting the next required motor command. This study confirms the possibility to sense and exploit the diverse soft body dynamics as the physical reservoir.

In summary, we have systemically investigated the five research questions concerning sensorimotor control of octopus-inspired soft robots.

8.1.1 Publications

Following the research conducted in this thesis, multiple papers have been published in peer-reviewed journals and related conferences [Li et al., 2011; Nakajima et al., 2011a; Nakajima et al., 2011b; Nakajima et al., 2011c; Nakajima et al., 2011d; Nakajima et al., 2011e; Li et al., 2012a; Li et al., 2012b; Nakajima et al., 2012; Li et al., 2013; Nakajima et al., 2013a; Nakajima et al., 2013b].

Results of the following papers are included in this thesis:

- Nakajima, K., Li, T., Hauser, H., Iida, F., and Pfeifer, R. (2013). Short-Term Memory in a Silicone-Based Soft Robotic Arm, Submitted to *Nature Communications*.

- Li, T., Nakajima, K., and Pfeifer, R. (2013). Online Learning Technique for Behavior Switching in a Soft Robotic Arm, In *the 2013 IEEE International Conference on Robotics and Automation*, pp. 1288–1294.

- Li, T., Nakajima, K., Calisti, M., Laschi, C., and Pfeifer, R. (2012). Octopus-Inspired Sensorimotor Control of a Multi-Arm Soft Robot, in *the 2012 IEEE International Conference on Mechatronics and Automation*, pp. 948–955.
- Li, T., Nakajima, K., Cianchetti, M., Laschi, C., and Pfeifer, R. (2012). Behavior Switching by Using Reservoir Computing for a Soft Robotic Arm, In *the 2012 IEEE International Conference on Robotics and Automation*, pp. 4918–4924.
- Li, T., Nakajima, K., Kuba, M., Gutnick, T., Hochner, B., and Pfeifer, R. (2011). From the Octopus to Soft Robot Control: An Octopus Inspired Behavior Control Architecture for Soft Robots, *Vie et Milieu/ Life and Environment*, 61 (4): pp. 211–217.
- Nakajima, K., Li, T., Kuppuswamy, N., and Pfeifer, R. (2011). Harnessing the Dynamics of a Soft Body with "Timing": Octopus Inspired Control via Recurrent Neural Networks, In *the 15th International Conference on Advanced Robotics*, pp. 277–284.

Papers which are not included in this thesis, but closely related to the theme of studying octopus-inspired soft robots:

- Li, T., Nakajima, K., and Pfeifer, R. (2013). Learning from the Octopus: Sensorimotor Control of Octopus-Inspired Soft Robots. In *the 2013 International Workshop on Soft Robotics and Morphological Computation*, p. 34.
- Nakajima, K., Li, T., Hauser, H., and Pfeifer, R. (2013). Muscular-Hydrostat Computers: Toward a Novel Control Scheme for Soft Robots, In *the 2013 International Workshop on Soft Robotics and Morphological Computation*, p. 49.
- Nakajima, K., Li, T., Kang, R., Guglielmino, E., Caldwell, D., and Pfeifer, R. (2012). Local Information Transfer in Soft Robotic Arm, in *the 2012 IEEE International Conference on Robotics and Biomimetics*, pp. 1273–1280.
- Nakajima, K., Li, T., Sumioka, H., Cianchetti, M., and Pfeifer, R. (2011). Information Theoretic Analysis on a Soft Robotic Arm Inspired by the Octopus, In *the 2011 IEEE International Conference on Robotics and Biomimetics*, pp. 110–117.
- Nakajima, K., Li, T., and Pfeifer, R. (2011). Timing and Behavioral Efficiency in Controlling a Soft Body: A Case Study in Octopus Reaching Behavior, In *the 2th International Conference on Morphological Computation*, pp. 132–134.
- Li, T., Nakajima, K., and Cianchetti, M. (2011). Finding Structure in Deadtime, In *the 2th International Conference on Morphological Computation*, pp. 47–49.
- Nakajima, K., Li, T., Kuppuswamy, N., and Pfeifer, R. (2011). Biologically Inspired Control of a Simulated Octopus Arm via Recurrent Neural Networks, in *the 13th annual conference companion on Genetic and evolutionary computation*, pp. 21–22.

- Nakajima, K., Li, T., Kuppuswamy, N., and Pfeifer, R. (2011). How to Harness the Dynamics of Soft Body: Timing Based Control of a Simulated Octopus Arm via Recurrent Neural Networks, *Procedia Computer Science*, 7: pp. 246–247.
- Kuppuswamy, N., Li, T., Nakajima, K., Cianchetti, M., and Pfeifer, R. (2009). A Biologically Inspired Approach to the Control of Octopus-like Soft Robot Arms, *Journal of Molecular Neuroscience*, 39 (Suppl 1): p. s65.

8.2 Implications

The results of this thesis and our experience during the investigation have some implications, both for the octopus biology and for the field of soft robotics.

8.2.1 Implications for the Octopus Biology

The investigations presented in this thesis provide some implications on the possible mechanisms that the octopus adopts to control its completely soft body.

Model the Octopus Brain as an ESN

Neurophysiological study on the organization of the octopus brain to examine how the brain represents a large number of movements suggests that there is no central topographical organization of the diverse movements, but motor circuits are distributed over the brain [Zullo et al., 2009]. Our implementation of the sensorimotor control scheme of octopus-inspired soft robots by the ESN has been demonstrated to be able to achieve the function of embedding and switching among multiple movements. Our studies provide a possible model of the octopus brain and implies that the octopus brain perhaps could be modeled as an ESN: visual and tactile information as the input and global parameters to initiate the motor programs locally embedded in the PNS as the output. The ESN has a useful property that the input can connect to any neurons in the reservoir. This property meets the observations that the same octopus behavior could be induced by local stimulation applied at any location throughout the basal lobe system [Zullo et al., 2009].

Multi-arm Coordination Mechanism

Most of the studies in this thesis (Chapter 3–5) have been devoted to the level of single-arm control; and only Chapter 6 investigated the multi-arm coordination mechanism. There are two reasons for our primary focus on the single-arm control instead of multi-arm coordination. First, the biological studies on the octopus have mainly centered on the single-arm level and; as a result, the single-arm control mechanism is relatively well-documented, as reviewed in Chapter 1. In contrast, the multi-arm coordination mechanism of the octopus has been rarely studied. Second,

building a multi-arm soft robot requires the accessibility of single soft robotic arms and thus has more technical challenges. Therefore, experiments on a multi-arm soft robot are more difficult and only become possible in a much later stage of our investigation.

So far, many studies on the multi-arm coordination mechanisms, especially in quadrupeds and hexapods, have been based on the concept of central pattern generators (CPGs) [Ijspeert, 2008] and reflex-based distributed neural networks [Espenschied et al., 1996; Watanabe et al., 2012]. We did not adopt these lines of investigations because of three considerations. First, biological findings do not support the CPG mechanism in the octopus motion generation [Flash and Hochner, 2005; Gutfreund et al., 2006]. Consequently, we eliminated the choice of CPG at the first place. Second, we have mainly focused on how the octopus determines the arm use in crawling based on visual information, which is well supported by biological investigations [Byrne et al., 2006b; Byrne et al., 2006a]. The control of an individual arm was assumed to be locally embedded in the arm PNS. However, this assumption does not exclude the possibility of a reflex-based distributed neural network control mechanism. Last, currently there is no evidence showing the octopus interbrachial commissure, a circular nerve structure among the arms, has the similar function of multi-arm coordination as other animals, for example a brittle star. Moreover, unlike the nervous system of a brittle star [Cobb and Stubbs, 1981; Stubbs and Cobb, 1981; Cobb and Stubbs, 1982; Watanabe et al., 2012], the octopus does has a brain. Based on these reasons, the multi-arm coordination mechanism proposed in this thesis has mainly focused on the distribution of control in the CNS, the PNS, and the body levels.

8.2.2 Implications for Soft Robot Design and Control

Our research on the octopus-inspired soft robots also provides some insights for soft robot design and control, in general.

Levels of Biomimicry

A long lasting debate in biomimetics is in which level of detail we should mimic the biological creatures [Benyus, 2002]. At the early stage of our study, we tried to mimic the natural form of the octopus arm by duplicating the muscular-hydrostat structure as closely as possible and build an octopus-inspired soft robot bottom-up from the tissue level. However, we soon realized the constraints of lacking suitable soft construction materials and actuators. The attempt to reproduce the propulsion and feeding motions of a jellyfish by other researchers has also demonstrated that directly copying the animal morphology could not produce the optimal performance of a soft robot [Nawroth et al., 2012]. A soft robot that faithfully mimics an animal structure may produce hypotheses that would be more readily accepted by the biology community, but soft robotics research should focus on a deeper level of biomimicry and study the natural processes and mechanisms because of the obvious lack of mature enabling technologies.

Proper Tasks for Soft Robots

Soft robots are characterized by slow body dynamics: there is an obvious time delay to transfer the applied forces through the body. As a result, aiming at precisely control the end effector of a soft robotic arm is equivalent to working against the nature. Soft robotics research should focus on tasks that are difficult for traditional rigid robots. For example, the control of a soft robotic grasper does not need to be precise or fast because soft materials can implicitly take over part of the control functions by adapting to objects of different shapes and surface properties [Ilievski et al., 2011].

Exploiting the Body Dynamics of Soft Robots

The rich body dynamics of a soft robot is created by the nonlinear interactions of sequences of slowly spreading activating forces. The rich body dynamics imposes an enormous challenge to traditional robotic control techniques, but can be exploited as a computational resource. Work by [Onal and Rus, 2013] demonstrates a case study to use the soft body dynamics to readily generate a mechanical sinusoidal output from a simple electrical square wave input. Our study in Chapter 7 indicates new possibilities of exploiting a soft body by showing its memory capacities. Therefore, a good soft robot design should welcome and exploit the softness, instead of constraining it.

8.3 Future Directions

As with any piece of research work, there are limitations left to be addressed and new possibilities that emerge during the investigation.

8.3.1 Investigating More Challenging Control Tasks

One limitation of our soft robotics platforms is that the behaviors can be tested are limited. In Chapters 4-5, only oscillatory arm movements of different frequencies are adopted. These kinds of oscillatory movements are frequently observed in octopus exploration and swimming behaviors. However, other interesting octopus behaviors, such as fetching, bipedal locomotion, and fine manipulation of object by suck-arm coordination, cannot be achieved and studied at our current platforms. The implementation of these behaviors on a soft robotic platform would bring new challenges and insights into the sensorimotor control of soft robots.

8.3.2 Factors Influencing the Computational Capacity of Soft Bodies

We demonstrated the possibility to use a soft robotic arm as a computational device in Chapter 7. Soft bodies with different construction materials and design parameters could have different computation potentials. For instance, a soft robotic arm can be built with silicone of different shore hardness and exhibit various richness of body dynamics under the same actuation. Moreover, the number of sensors embedded in a soft body to detect the dynamics also can be investigated. We speculate that a larger number of sensors, until a certain limit, would represent the body dynamics in higher dimensions and benefit the computation capacity. Other factors could influence the computational power of a soft body include but not limit to: the spatial arrangement of sensors, the nonlinearity of sensor response, the properties of medium in which the arm moves and so on. These factor would be studied in detail in our future work.

8.3.3 New Soft Robot Actuators

The capability of our soft robotic platform is currently constrained by the limited choice of reliable and conveniently accessible soft robot actuators. Some actuators frequently used for soft robots were excluded in our studies based on preliminary tests. For example, electroactive polymer [Bar-Cohen, 2000] based actuators were not selected because they require applying a high voltage up to few thousands Volts and can only last for a quite limited number of actuation cycles. Our learning-based approach requires extensive experimental exploration of the control parameter space and frequent interaction between the robot and human, so we used tendon-driven mechanisms powered by servo motors. Because both the motors and cable-guiding mechanisms are difficult to be minimized, this actuation method ultimately limited the number of actuators that could be included in our soft robotic platform. Miniature hydraulic and pneumatic actuators [De Volder and Reynaerts, 2010] seem to be a promising actuation choice. For underwater soft robots, combining miniature hydraulic devices and water as the hydraulic fluid [Addae-mensah et al., 2013] would endow soft robots with more independent control variables and eliminate the usual constraint of the need of a large tank to hold hydraulic fluid. For over-ground and aerial soft robots, the combination of miniature pneumatic devices and on-bard power generation [Onal et al., 2011] would be a favorable solution.

Besides these relatively traditional actuators, there are some emerging technologies which have the potential to bring revolutionary approaches to soft robotics research. Bio-hybrid actuators [Nawroth et al., 2012; Baryshyan et al., 2012; Ricotti and Menciassi, 2012], which are built mainly from biopolymers and living tissues, would be increasingly used to actuate soft robots. These interdisciplinary approaches call for intensive collaboration among multiple research fields, especially robotics, neuroscience, material science, and biology.

Bibliography

[Addae-mensah et al., 2013] Addae-mensah, K., Cheung, Y. K., and Sia, S. K. (2013). Microfluidic flow devices, methods and systems. US Patent 20,130,048,092.

[Albu-Schaeffer et al., 2009] Albu-Schaeffer, A., Eiberger, O., Fuchs, M., Grebenstein, M., Haddadin, S., Ott, C., Stemmer, A., Wimböck, T., Wolf, S., Borst, C., and et al. (2009). Anthropomorphic soft robotics - from torque control to variable intrinsic compliance. In *the 2009 International Symposium on Robotics Research*, pages 185–207.

[Albu-Schaeffer et al., 2008] Albu-Schaeffer, A.ffer, A., Eiberger, O., Grebenstein, M., Haddadin, S., Ott, C., Wimbock, T., Wolf, S., and Hirzinger, G. (2008). Soft robotics. *IEEE Robotics & Automation Magazine*, 15(3):20–30.

[Altman, 1971] Altman, J. (1971). Control of accept and reject reflexes in the octopus. *Nature*, 229:204–206.

[Bar-Cohen, 2000] Bar-Cohen, Y. (2000). Electroactive polymers as artificial muscles-capabilities, potentials and challenges. *Handbook on biomimetics*, 11(8):1–3.

[Baryshyan et al., 2012] Baryshyan, A. L., Woods, W., Trimmer, B. A., and Kaplan, D. L. (2012). Isolation and maintenance-free culture of contractile myotubes from manduca sexta embryos. *PLoS ONE*, 7(2):e31598.

[Beer et al., 2009] Beer, F. P., Johnston, E. R., Dewolf, J. T., and Mazurek, D. F. (2009). *Mechanics of materials*. McGraw-Hill, 5th edition.

[Beer et al., 1997] Beer, R. D., Quinn, R. D., Chiel, H. J., and Ritzmann, R. E. (1997). Biologically inspired approaches to robotics: what can we learn from insects? *Communications of the ACM*, 40(3):30–38.

[Benyus, 2002] Benyus, J. M. (2002). *Biomimicry: Innovation Inspired by Nature*. William Morrow Paperbacks, 2nd edition.

[Brown et al., 2010] Brown, E., Rodenberg, N., Amend, J., Mozeika, A., Steltz, E., Zakin, M. R., Lipson, H., and Jaeger, H. M. (2010). Universal robotic gripper based on the jamming of granular material. *Proceedings of the National Academy of Sciences*, 107:18809–18814.

[Byrne et al., 2006a] Byrne, R. A., Kuba, M. J., Meisel, D. V., Griebel, U., and Mather, J. A. (2006a). Does octopus vulgaris have preferred arms? *Journal of Comparative Psychology*, 120(3):198–204.

[Byrne et al., 2006b] Byrne, R. A., Kuba, M. J., Meisel, D. V., Griebel, U., and Mather, J. A. (2006b). Octopus arm choice is strongly influenced by eye use. *Behavioural Brain Research*, 172(2):195–201.

[Calisti et al., 2011] Calisti, M., Giorelli, M., Levy, G., Mazzolai, B., Hochner, B., Laschi, C., and Dario, P. (2011). An octopus-bioinspired solution to movement and manipulation for soft robots. *Bioinspiration & Biomimetics*, 6(3):036002.

[Chirikjian and Burdick, 1991] Chirikjian, G. and Burdick, J. (1991). Hyper-redundant robot mechanisms and their applications. In *the 1991 IEEE/RSJ International Workshop on Intelligent Robots and Systems*, volume 1, pages 185–190.

[Chirikjian, 1992] Chirikjian, G. S. (1992). *Theory and Applications of Hyper Redundant Robotic Manipulators*. PhD thesis, California Institute of Technology, Department of Applied Mathmatics.

[Chirikjian and Burdick, 1994] Chirikjian, G. S. and Burdick, J. W. (1994). Modal approach to hyper-redundant manipulator kinematics. *IEEE Transactions on Robotics and Automation*, 10(3):343–354.

[Cobb and Stubbs, 1981] Cobb, J. and Stubbs, T. (1981). The giant neurone system in ophiuroids i. the general morphology of the radial nerve cords and circumoral nerve ring. *Cell and Tissue Research*, 219(1):197–207.

[Cobb and Stubbs, 1982] Cobb, J. and Stubbs, T. (1982). The giant neurone system in ophiuroids iii. the detailed connections of the circumoral nerve ring. *Cell and Tissue Research*, 226(3):675–687.

[Crane, 2008] Crane, C. D. D. J. (2008). *Kinematic analysis of robot manipulators*. Cambridge University Press.

[De Volder and Reynaerts, 2010] De Volder, M. and Reynaerts, D. (2010). Pneumatic and hydraulic microactuators: a review. *Journal of Micromechanics and microengineering*, 20(4):043001.

[Dowling, 1997] Dowling, K. J. (1997). *Limbless Locomotion: Learning to Crawl with a Snake Robot*. PhD thesis, Carnegie Mellon University, The Robotics Institute.

[Edelman, 2007] Edelman, G. M. (2007). Learning in and from brain-based devices. *Science*, 318(5853):1103–1105.

[Espenschied et al., 1996] Espenschied, K. S., Quinn, R. D., Beer, R. D., and Chiel, H. J. (1996). Biologically based distributed control and local reflexes improve rough terrain locomotion in a hexapod robot. *Robotics and Autonomous Systems*, 18(1):59–64.

[Filippini et al., 2008] Filippini, R., Sen, S., and Bicchi, A. (2008). Toward soft robots you can depend on. *IEEE Robotics & Automation Magazine*, 15(3):31–41.

[Finn et al., 2009] Finn, J. K., Tregenza, T., and Norman, M. D. (2009). Defensive tool use in a coconut-carrying octopus. *Current Biology*, 19(23):R1069–R1070.

[Flash and Hochner, 2005] Flash, T. and Hochner, B. (2005). Motor primitives in vertebrates and invertebrates. *Current Opinion in Neurobiology*, 15(6):660–666.

[Gray, 1970] Gray, E. G. (1970). The fine structure of the vertical lobe of octopus brain. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, pages 379–394.

[Graziadei, 1971] Graziadei, P. (1971). The nervous system of the arms. *The Anatomy of the Nervous System of Octopus vulgaris. Young JZ pp44-61, Clarendon Press, Oxford*.

[Gutfreund, 1998] Gutfreund, Y. (1998). Patterns of arm muscle activation involved in octopus reaching movements. *Journal of Neuroscience*, 18(15):5976–5987.

[Gutfreund et al., 1996] Gutfreund, Y., Flash, T., Yarom, Y., Fiorito, G., Segev, I., and Hochner, B. (1996). Organization of octopus arm movements: A model system for studying the control of flexible arms. *Journal of Neuroscience*, 16(22):7297–7307.

[Gutfreund et al., 2006] Gutfreund, Y., Matzner, H., Flash, T., and Hochner, B. (2006). Patterns of motor activity in the isolated nerve cord of the octopus arm. *The Biological Bulletin*, 211(3):212–222.

[Gutnick et al., 2011] Gutnick, T., Byrne, R., Hochner, B., and Kuba, M. (2011). Octopus vulgaris uses visual information to determine the location of its arm. *Current Biology*, 21(6):460–462.

[Halloy et al., 2007] Halloy, J., Sempo, G., Caprari, G., Rivault, C., Asadpour, M., Tache, F., Said, I., Durier, V., Canonge, S., Ame, J., et al. (2007). Social integration of robots into groups of cockroaches to control self-organized choices. *Science*, 318(5853):1155–1158.

[Hammer et al., 2009] Hammer, B., Schrauwen, B., and Steil, J. J. (2009). Recent advances in efficient learning of recurrent networks. In *the 17th European Symposium on Artificial Neural Networks*, pages 213–226.

[Hannan and Walker, 2003] Hannan, M. W. and Walker, I. D. (2003). Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots. *Journal of Robotic Systems*, 20(2):45–63.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *Linear Methods for Regression*. Springer.

[Hauser et al., 2011] Hauser, H., Ijspeert, A., Fuechslin, R., Pfeifer, R., and Maass, W. (2011). Towards a theoretical foundation for morphological computation with compliant bodies. *Biological Cybernetics*, 105(5-6):355–370.

[Hauser et al., 2012] Hauser, H., Ijspeert, A., Fuechslin, R., Pfeifer, R., and Maass, W. (2012). The role of feedback in morphological computation with compliant bodies. *Biological Cybernetics*, 106(10):595–613.

[Haykin, 2008] Haykin, S. O. (2008). *Neural networks and learning machines*. Prentice Hall, 3rd edition.

[Hopkins et al., 2009] Hopkins, J. K., Spranklin, B. W., and Gupta, S. K. (2009). A survey of snake-inspired robot designs. *Bioinspiration & Biomimetics*, 4(2):021001.

[Huffard, 2006] Huffard, C. L. (2006). Locomotion by abdopus aculeatus (cephalopoda: Octopodidae): walking the line between primary and secondary defenses. *Journal of Experimental Biology*, 209(19):3697–3707.

[Huffard et al., 2005] Huffard, C. L., Boneka, F., and Full, R. J. (2005). Underwater bipedal locomotion by octopuses in disguise. *Science*, 307(5717):1927–1927.

[Hughes et al., 1991] Hughes, P., Sincarsin, W., and Carroll, K. (1991). Trussarm—a variable-geometry-truss manipulator. *Journal of Intelligent Material Systems and Structures*, 2(2):148–160.

[Iida and Laschi, 2011] Iida, F. and Laschi, C. (2011). Soft robotics: Challenges and perspectives. *Procedia Computer Science*, 7(0):99–102.

[Ijspeert, 2008] Ijspeert, A. J. (2008). Central pattern generators for locomotion control in animals and robots: a review. *Neural Networks*, 21(4):642–653.

[Ijspeert and Crespi, 2007] Ijspeert, A. J. and Crespi, A. (2007). Online trajectory generation in an amphibious snake robot using a lamprey-like central pattern generator model. In *the 2007 IEEE International Conference on Robotics and Automation*, pages 262–268.

[Ijspeert et al., 2007] Ijspeert, A. J., Crespi, A., Ryczko, D., and Cabelguen, J.-M. (2007). From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420.

[Ilievski et al., 2011] Ilievski, F., Mazzeo, A. D., Shepherd, R. F., Chen, X., and Whitesides, G. M. (2011). Soft robotics for chemists. *Angewandte Chemie International Edition*, page 1890–1895.

[Jaeger, 2001] Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. Technical Report 148, German National Research Center for Information Technology.

[Jaeger, 2002] Jaeger, H. (2002). Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. Technical Report 159, German National Research Center for Information Technology.

[Jaeger and Haas, 2004] Jaeger, H. and Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80.

[Jones and Walker, 2006] Jones, B. and Walker, I. (2006). Kinematics for multisection continuum robots. *IEEE Transactions on Robotics*, 22(1):43–55.

[Kier and Smith, 1985] Kier, W. M. and Smith, K. K. (1985). Tongues, tentacles and trunks: the biomechanics of movement in muscular-hydrostats. *Zool. J. Linn. Soc.*, 83(4):307–324.

[Kim et al., 2013] Kim, S., Laschi, C., and Trimmer, B. (2013). Soft robotics: a bioinspired evolution in robotics. *Trends in Biotechnology*, 31(5):287–294.

[Krieger et al., 2000] Krieger, M. J., Billeter, J.-B., and Keller, L. (2000). Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995.

[Li et al., 2012a] Li, T., Nakajima, K., Calisti, M., Laschi, C., and Pfeifer, R. (2012a). Octopus-inspired sensorimotor control of a multi-arm soft robot. In *the 2012 International Conference on Mechatronics and Automation*, pages 948–955.

[Li et al., 2012b] Li, T., Nakajima, K., Cianchetti, M., Laschi, C., and Pfeifer, R. (2012b). Behavior switching by using reservoir computing for a soft robotic arm. In *the 2012 IEEE International Conference on Robotics and Automation*, pages 4918–4924.

[Li et al., 2011] Li, T., Nakajima, K., Kuba, M., Gutnick, T., Hochner, B., and Pfeifer, R. (2011). From the octopus to soft robots control: an octopus inspired behavior control architecture for soft robots. *Vie et Milieu/ Life and Environment*, 61 (4):211–217.

[Li et al., 2013] Li, T., Nakajima, K., and Pfeifer, R. (2013). Online learning technique for behavior switching in a soft robotic arm. In *the 2013 IEEE International Conference on Robotics and Automation*, pages 1288–1294.

[Lin et al., 2011] Lin, H.-T., Leisk, G. G., and Trimmer, B. (2011). GoQBot: a caterpillar-inspired soft-bodied rolling robot. *Bioinspiration & Biomimetics*, 6(2):026007.

[Lukosevicius and Jaeger, 2009] Lukosevicius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.

[Maass et al., 2002] Maass, W., Natschlaeger, T., and Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560.

[MacKerrow, 1995] MacKerrow, P. J. (1995). *Introduction to robotics*. Sydney: Addison-Wesley.

[Maeda et al., 2007] Maeda, S., Hara, Y., Yoshida, R., and Hashimoto, S. (2007). Chemical robot—design of self-walking gel—. In *the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2150–2155.

[Majidi, 2013] Majidi, C. (2013). Soft robotics: A perspective—current trends and prospects for the future. *Soft Robotics*, 1(P):5–11.

[Mather, 1998] Mather, J. A. (1998). How do octopuses use their arms? *Journal of Comparative Psychology*, 112(3):306–316.

[Meier et al., 2005] Meier, P., Lang, M., and Oberthuer, S. (2005). Reiterated tension testing of silicone elastomer. *Plastics, Rubber and Composites*, 34:372–377.

[Mochiyama et al., 1998] Mochiyama, H., Shimemura, E., and Kobayashi, H. (1998). Shape correspondence between a spatial curve and a manipulator with hyper degrees of freedom. In *the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 161–166.

[Mochiyama and Suzuki, 2003] Mochiyama, H. and Suzuki, T. (2003). Kinematics and dynamics of a cable-like hyper-flexible manipulator. In *the 2003 IEEE International Conference on Robotics and Automation*, volume 3, pages 3672–3677.

[Nakajima et al., 2013a] Nakajima, K., Li, T., Hauser, H., Iida, F., and Pfeifer, R. (2013a). Short-term memory in a silicone-based soft robotic arm. Submitted for publication.

[Nakajima et al., 2013b] Nakajima, K., Li, T., Hauser, H., and Pfeifer, R. (2013b). Muscular-hydrostat computers: Toward a novel control scheme for soft robots. In *the 2013 International Workshop on Soft Robotics and Morphological Computation*, page 49.

[Nakajima et al., 2012] Nakajima, K., Li, T., Kang, R., Guglielmino, E., Caldwell, D. G., and Pfeifer, R. (2012). Local information transfer in soft robotic arm. In *the 2012 IEEE International Conference on Robotics and Biomimetics*, pages 1273–1280.

[Nakajima et al., 2011a] Nakajima, K., Li, T., Kuppuswamy, N., and Pfeifer, R. (2011a). Biologically inspired control of a simulated octopus arm via recurrent neural networks. In *the 13th annual conference companion on Genetic and evolutionary computation (GECCO '11)*, pages 21–22.

[Nakajima et al., 2011b] Nakajima, K., Li, T., Kuppuswamy, N., and Pfeifer, R. (2011b). Harnessing the dynamics of a soft body with "timing": Octopus inspired control via recurrent neural networks. In *the 15th International Conference on Advanced Robotics*, pages 277–284.

[Nakajima et al., 2011c] Nakajima, K., Li, T., Kuppuswamy, N., and Pfeifer, R. (2011c). How to harness the dynamics of soft body: Timing based control of a simulated octopus arm via recurrent neural networks. *Procedia Computer Science*, 7(0):246–247.

[Nakajima et al., 2011d] Nakajima, K., Li, T., and Pfeifer, R. (2011d). Timing and behavioral efficiency in controlling a soft body: A case study in octopus reaching behavior. In *the 2th International Conference on Morphological Computation (ICMC)*, pages 132– 134.

[Nakajima et al., 2011e] Nakajima, K., Li, T., Sumioka, H., Cianchetti, M., and Pfeifer, R. (2011e). Information theoretic analysis on a soft robotic arm inspired by the octopus. In *the 2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 110–117.

[Nawroth et al., 2012] Nawroth, J. C., Lee, H., Feinberg, A. W., Ripplinger, C. M., McCain, M. L., Grosberg, A., Dabiri, J. O., and Parker, K. K. (2012). A tissue-engineered jellyfish with biomimetic propulsion. *Nature Biotechnology*, 30(8):792–797.

[Onal et al., 2011] Onal, C. D., Chen, X., Whitesides, G. M., and Rus, D. (2011). Soft mobile robots with on-board chemical pressure generation. In *the 2011 International Symposium on Robotics Research*.

[Onal and Rus, 2013] Onal, C. D. and Rus, D. (2013). Autonomous undulatory serpentine locomotion utilizing body dynamics of a fluidic soft robot. *Bioinspiration & Biomimetics*, 8(2):026003.

[Otake et al., 2002] Otake, M., Kagami, Y., Inaba, M., and Inoue, H. (2002). Motion design of a starfish-shaped gel robot made of electro-active polymer gel. *Robotics and Autonomous Systems*, 40(2-3):185–191.

[Packard, 1972] Packard, A. (1972). Cephalopods and fish: the limits of convergence. *Biological Reviews*, 47(2):241–307.

[Paul, 2006] Paul, C. (2006). Morphological computation: a basis for the analysis of morphology and control requirements. *Robotics and Autonomous Systems*, 54(8):619–630.

[Pfeifer and Bongard, 2006] Pfeifer, R. and Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

[Pfeifer et al., 2007] Pfeifer, R., Lungarella, M., and Iida, F. (2007). Self-organization, embodiment, and biologically inspired robotics. *Science*, 318(5853):1088–1093.

[Pfeifer et al., 2012] Pfeifer, R., Lungarella, M., and Iida, F. (2012). The challenges ahead for bio-inspired 'soft' robotics. *Communications of the ACM*, 55:76–87.

[Pfeifer and Scheier, 1999] Pfeifer, R. and Scheier, C. (1999). *Understanding Intelligence*. MIT Press.

[Ricotti and Menciassi, 2012] Ricotti, L. and Menciassi, A. (2012). Bio-hybrid muscle cell-based actuators. *Biomedical Microdevices*, 14(6):987–998.

[Rowell, 1963] Rowell, C. H. F. (1963). Excitatory and inhibitory pathways in the arm of octopus. *Journal of Experimental Biology*, 40(2):257–270.

[Rowell, 1966] Rowell, C. H. F. (1966). Activity of interneurones in the arm of octopus in response to tactile stimulation. *Journal of Experimental Biology*, 44(3):589–605.

[Sadd, 2009] Sadd, M. H. (2009). *Elasticity: theory, applications, and numerics*. Academic Press, 2nd edition.

[Shepherd et al., 2011] Shepherd, R. F., Ilievski, F., Choi, W., Morin, S. A., Stokes, A. A., Mazzeo, A. D., Chen, X., Wang, M., and Whitesides, G. M. (2011). Multigait soft robot. *Proceedings of the National Academy of Sciences*, 108(51):20400–20403.

[Shomrat et al., 2011] Shomrat, T., Graindorge, N., Bellanger, C., Fiorito, G., Loewenstein, Y., and Hochner, B. (2011). Alternative sites of synaptic plasticity in two homologous fan-out fan-in learning and memory networks. *Current Biology*, 21(21):1773–1782.

[Steil, 2004] Steil, J. J. (2004). Backpropagation-decorrelation: Online recurrent learning with o (n) complexity. In *the 2004 IEEE International Joint Conference on Neural Networks*, volume 2, pages 843–848.

[Steltz et al., 2009] Steltz, E., Mozeika, A., Rodenberg, N., Brown, E., and Jaeger, H. M. (2009). JSEL: Jamming skin enabled locomotion. In *the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5672–5677.

[Stubbs and Cobb, 1981] Stubbs, T. and Cobb, J. (1981). The giant neurone system in ophiuroids ii. the hyponeural motor tracts. *Cell and Tissue Research*, 220(2):373–385.

[Sugiyama and Hirai, 2006] Sugiyama, Y. and Hirai, S. (2006). Crawling and jumping by a deformable robot. *The International Journal of Robotics Research*, 25(5-6):603–620.

[Sumbre et al., 2005] Sumbre, G., Fiorito, G., Flash, T., and Hochner, B. (2005). Neurobiology: Motor control of flexible octopus arms. *Nature*, 433(7026):595–596.

[Sumbre et al., 2006] Sumbre, G., Fiorito, G., Flash, T., and Hochner, B. (2006). Octopuses use a human-like strategy to control precise point-to-point arm movements. *Current Biology*, 16(8):767–772.

[Sumbre et al., 2001] Sumbre, G., Gutfreund, Y., Fiorito, G., Flash, T., and Hochner, B. (2001). Control of octopus arm extension by a peripheral motor program. *Science*, 293(5536):1845–1848.

[Suzumori et al., 2007] Suzumori, K., Endo, S., Kanda, T., Kato, N., and Suzuki, H. (2007). A bending pneumatic rubber actuator realizing soft-bodied manta swimming robot. In *the 2007 IEEE International Conference on Robotics and Automation*, pages 4975–4980.

[Suzumori et al., 1991] Suzumori, K., Iikura, S., and Tanaka, H. (1991). Development of flexible microactuator and its applications to robotic mechanisms. In *the 1991 IEEE International Conference on Robotics and Automation*, volume 2, pages 1622–1627.

[Swevers et al., 2007] Swevers, J., Verdonck, W., and De Schutter, J. (2007). Dynamic model identification for industrial robots. *IEEE Control Systems Magazine*, 27(5):58–71.

[Tedrake, 2009] Tedrake, R. (2009). Underactuated robotics: Learning, planning, and control for efficient and agile machines. Course Notes for MIT 6.832.

[Thomson Reuters, 2013] Thomson Reuters (2013). Web of knowledge. `http://wcs.webofknowledge.com`.

[Trivedi et al., 2008] Trivedi, D., Rahn, C. D., Kier, W. M., and Walker, I. D. (2008). Soft robotics: Biological inspiration, state of the art, and future research. *Applied Bionics and Biomechanics*, 5(3):99 – 117.

[Umedachi et al., 2010] Umedachi, T., Takeda, K., Nakagaki, T., Kobayashi, R., and Ishiguro, A. (2010). Fully decentralized control of a soft-bodied robot inspired by true slime mold. *Biological Cybernetics*, 102(3):261–269.

[Umedachi et al., 2011] Umedachi, T., Takeda, K., Nakagaki, T., Kobayashi, R., and Ishiguro, A. (2011). A soft deformable amoeboid robot inspired by plasmodium of true slime mold. *International Journal of Unconventional Computing*, 7(6):449–462.

[Verstraeten et al., 2007] Verstraeten, D., Schrauwen, B., D'Haene, M., and Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403.

[Watanabe et al., 2012] Watanabe, W., Kano, T., Suzuki, S., and Ishiguro, A. (2012). A decentralized control scheme for orchestrating versatile arm movements in ophiuroid omnidirectional locomotion. *Journal of The Royal Society Interface*, 9(66):102–109.

[Webb, 2000] Webb, B. (2000). What does robotics offer animal behaviour? *Animal Behaviour*, 60(5):545–558.

[Webb, 2002] Webb, B. (2002). Robots in invertebrate neuroscience. *Nature*, 417(6886):359–363.

[Wells, 1960] Wells, M. (1960). Proprioception and visual discrimination of orientation in octopus. *Journal of Experimental Biology*, 37(3):489–499.

[Wells, 1964] Wells, M. J. (1964). Tactile discrimination of surface curvature and shape by the octopus. *Journal of Experimental Biology*, 41(2):433–445.

[Wells, 1978] Wells, M. J. (1978). *Octopus: physiology and behaviour of an advanced invertebrate*. Chapman and Hall, London; New York.

[Werbos, 1988] Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356.

[Williams and Zipser, 1989] Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

[Yekutieli et al., 2007] Yekutieli, Y., Mitelman, R., Hochner, B., and Flash, T. (2007). Analyzing octopus movements using three-dimensional reconstruction. *Journal of Neurophysiology*, 98(3):1775–1790.

[Yekutieli et al., 2005a] Yekutieli, Y., Sagiv-Zohar, R., Aharonov, R., Engel, Y., Hochner, B., and Flash, T. (2005a). Dynamic model of the octopus arm. i. biomechanics of the octopus reaching movement. *Journal of Neurophysiology*, 94(2):1443–1458.

[Yekutieli et al., 2005b] Yekutieli, Y., Sagiv-Zohar, R., Hochner, B., and Flash, T. (2005b). Dynamic model of the octopus arm. ii. control of reaching movements. *Journal of Neurophysiology*, 94(2):1459–1468.

[Yekutieli et al., 2002] Yekutieli, Y., Sumbre, G., Flash, T., and Hochner, B. (2002). How to move with no rigid skeleton? the octopus has the answers. *Biologist (London)*, 49(6):250–254.

[Young, 1965] Young, J. Z. (1965). The centres for touch discrimination in octopus. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 249(755):45–67.

[Young, 1971] Young, J. Z. (1971). *The anatomy of the nervous system of Octopus vulgaris*. Clarendon Press, Oxford.

[Zullo et al., 2009] Zullo, L., Sumbre, G., Agnisola, C., Flash, T., and Hochner, B. (2009). Nonsomatotopic organization of the higher motor centers in octopus. *Current Biology*, 19(19):1632–1636.

Appendix A

From the Octopus to Soft Robot Control: Octopus Inspired Behavior Control Architecture for Soft Robots

Reprinted from:

Li, T., Nakajima, K., Kuba, M., Gutnick, T., Hochner, B., and Pfeifer, R. (2011). From the Octopus to Soft Robot Control: An Octopus Inspired Behavior Control Architecture for Soft Robots, *Vie et Milieu/ Life and Environment*, 61 (4): pp. 211–217.

This is the final accepted version. The original publication is available at: http://www.obs-banyuls.fr/Viemilieu/index.php/volume-61-2011/61-issue-4/614-article-6.html

FROM THE OCTOPUS TO SOFT ROBOTS CONTROL: AN OCTOPUS INSPIRED BEHAVIOR CONTROL ARCHITECTURE FOR SOFT ROBOTS

T. LI[1], K. NAKAJIMA[1], M. KUBA[2,3], T. GUTNICK[2], B. HOCHNER[2,3], R. PFEIFER[1]*

[1] University of Zurich, Department of Informatics, 8050 Zurich, Switzerland
[2] Hebrew University, Institute of Life Sciences, Department of Neuroscience, Jerusalem 91904, Israel
[3] Hebrew University, Interdisciplinary Center for Neural Computation, Jerusalem 91904, Israel
** Corresponding author: taoli@ifi.uzh.ch*

SOFT ROBOTICS
OCTOPUS
BIO-INSPIRED CONTROL
EMBODIMENT

ABSTRACT. – In recent years, breakthroughs have been made in both biology and robotics by the close cooperation between biologists and roboticists. Researchers in the fields of octopus biological study and soft robotics can also benefit from working together and inspiring each other. However, this collaboration is not easy because of the different research motivations and terminologies used. This paper starts from challenges for controlling soft robots, through biological inspirations from the octopus, to their engineering interpretation for creating a behavior control architecture for soft robots. Hypotheses related to the octopus biology from the engineering perspective are also proposed. This work serves as a case study in biologist-roboticist collaboration. It wishes to promote mutual understanding and cooperation between these two disciplines.

INTRODUCTION

It has become common practice for roboticists and engineers in general to search for engineering solutions from biological studies (Bekey 2005). Meanwhile, there is a growing body of robotics research that serves as testbeds for biological hypotheses and aids in developing novel ones (Webb 2000, 2001, 2002, Delcomyn 2004). One such example is the development of a salamander robot, created on the base of neurophysiological study of the animal (Ijspeert *et al*. 2007). The robot matches the biological finding that the transition between walking and swimming can be produced by purely changing the strength of the brain signal which drives locomotion (Cabelguen *et al*. 2003). The study also predicted that the body oscillatory centers have a higher intrinsic frequency than the limb ones, a hypothesis that was supported by later biological research (Ijspeert *et al*. 2007).

Soft robotics represents one of the promising, yet challenging, trends in biologically inspired robotics. Traditional robots resemble the limb structures of vertebrates or arthropods and are constructed mainly of rigid materials. Motivated by the fact that soft material is ubiquitous in the body structure of living creatures, a new family of robots aim to involve flexible elements in the construction. In this study, we take the extreme case and consider soft robots as robotic arms that are constructed by soft materials exclusively. In terms of morphological flexibility and interaction safety, they have significant advantages over traditional articulated robots. The flexibility gives soft robots the potential to be used as search and rescue robots, which could crawl through rubble and squeeze into small spaces. Accompanying these benefits are enormous challenges in controlling them (Trivedi *et al*. 2008). We consider a few of these control difficulties in this paper. Firstly, soft robots contain unlimited degrees of freedom (DoFs). To control the robots by regulating individual DoF is computationally expensive since it needs unlimited numbers of control parameters. Secondly, the structures of soft robots are characterized by large deformation under normal working conditions. The deformation is non-negligible compared with their body sizes. This characteristic violates the small deformation assumption of the traditional kinematic and dynamic model-based robotic control methods (MacKerrow 1995). Moreover, soft robots exhibit complex dynamics because of their deformable bodies (Nakajima *et al*. 2011a, 2011b, 2011c). Force redistribution in the structure caused by local perturbation is difficult to predict (Nie *et al*. 2009). Thus, it is hard, if not impossible, to explicitly derive equations to quantify and control soft robots. Facing these difficulties, there is still no efficient method tailored for controlling soft robots.

By summarizing and interpreting the biological studies on octopus anatomy, behavior, and neurophysiology and following the framework of embodiment, we propose in this paper a novel octopus inspired behavior control architecture for soft robots. Robot behavior control architecture integrates sensory inputs and generates appropriate motor outputs. The octopus is an ideal source of inspiration for controlling soft robots because of its remarkable abilities to manage its flexible arms to perform various movements (Gutfreund *et al*. 1996, Mather 1998, Sumbre *et al*. 2001, Flash & Hochner 2005, Gutnick *et al*. 2011a,

2011b), including precise point-to-point fetching (Sumbre *et al*. 2005, 2006) in the unstructured water environment. Embodiment is a fundamental concept in embodied artificial intelligence, neuroscience, and cognitive science (Brooks 1991a, 1991b, Edelman 2007). Its core claim is that the behaviors of robots and animals are not only the result of control from nervous (control) systems, but also of their body properties and interaction with the environment (Pfeifer & Scheier 1999, Pfeifer & Bongard 2006, Pfeifer *et al*. 2007).

This paper intends to formulate a synthesis of octopus inspired behavior control architecture for soft robots. Nevertheless, the detailed implementation of the control architecture as well as quantitative comparison between the performance of the octopus and soft robots equipped with the proposed control architecture is out of the scope. It also reports the principles and methods applied to stimulate discussion about good practices for this type of cross-disciplinary study. The contributions of this work are three-fold. First, it proposes a novel behavior control architecture to the intrinsic challenges in controlling soft robots. Second, some biological hypotheses are proposed from the engineering point of view. Last, by showing how such research might benefit both sides, it wishes to promote mutual understanding and cooperation between biologists and roboticists.

METHODS

The prerequisite of the developed soft robots behavior control architecture is that it should not only contribute new insights to the robotics field, but also has the possibility to be used to test biological hypotheses. There are five approaches and guidelines that were followed:

i. By interpreting biological literature and direct communication between biologists and roboticists. This is the most straightforward and commonly adopted way in biologically inspired robot projects. In this study, a consortium of octopus researchers and robot engineers works closely together to translate biological studies to engineering solutions.

ii. If sufficient biological findings are available, roboticists interpret them by feasible engineering implementation.

iii. For questions that cannot be clearly answered by existing octopus biological findings, assumptions about the biological mechanisms are made jointly by biologists and roboticists. These assumptions not only pave a way for engineers but could also be further tested by biologists.

iv. The robot control architecture should be biologically plausible in order to have the possibility to serve as a tool to test octopus biological hypotheses.

v. The resulting control architecture is evaluated by a 3D soft robot physical simulator. Correspondingly, unsatisfactory assumptions are rejected and replaced by new ones.

The soft robots behavior control architecture, which is implemented in Java, runs on a regular PC.

RESULTS

The initial challenge to be solved in this section is dealing with unlimited DoFs and complex body dynamics. A soft robot physical simulator used to facilitate evaluating the control architecture is introduced first. Next, a timing based control and its engineering implementation are presented to deal with the control challenges. Finally, we incorporate implications from embodiment and synthesize a novel octopus inspired soft robot behavior control architecture.

Soft robot physical simulator

The soft robot physical simulator is shown in Fig. 1. It is an extension of the 2D model proposed by Yekutieli *et*

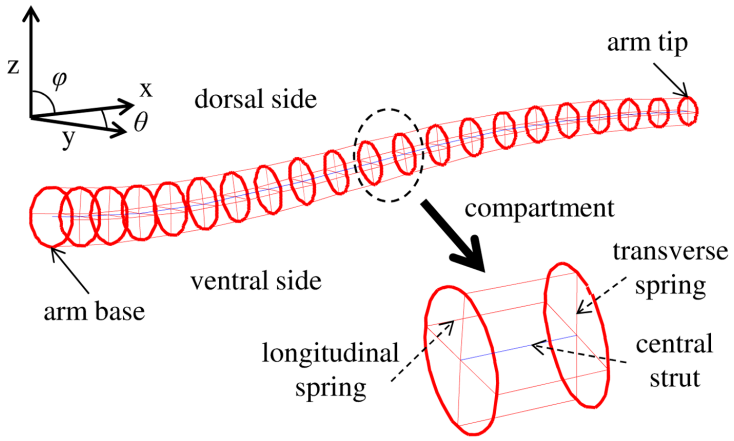


Fig. 1. – A 3D octopus arm physical simulator. It consists of 20 segments made of massless springs, a central strut, and point masses. The arm motion is governed by the forces acting on each point mass.

al. (2005a). The simulator consists of point masses connected by massless springs (muscles). It has 20 compartments, which maintain constant volume to simulate the muscular hydrostat property. Each compartment contains four longitudinal muscles, four transverse muscles, a central strut, and eight point masses. The positions of point masses are determined by the net forces acted on them. Details of the simulator construction are reported by Kang *et al*. (2011).

Control infinite DoFs

The infinite joint space dimensions of soft arms contribute the manipulation flexibility, but also impose a motor control challenge in engineering because of the huge number of control parameters. This problem is solved by directly interpreting biological studies since the underlying mechanisms have been clearly proposed and can be implemented in engineering in a straightforward manner.

The overall control architecture is organized hierarchically and consists of two levels, inspired by biological studies showing that the octopus central nervous system (CNS) sends a limited number of global parameters to its peripheral nervous system (PNS) to control its arms for goal-oriented voluntary movements (Gutfreund *et al*. 1996, 2006, Gutfreund 1998, Sumbre *et al*. 2001). Compared with traditional hierarchical controllers in engineering, this control architecture is unique in the way that the control functions are highly distributed. The high-level controller only initiates motions, while the low-level controller assumes a large part of control functions and is greatly autonomous. The distributed control strategy and highly autonomous low-level controller are motivated by that the octopus PNS which contains roughly two thirds of the neurons of the animal's nervous system (Young 1963, 1971, Hochner *et al*. 2006) is highly autonomous. Both the reaching (Gutfreund *et al*. 1996, Gutfreund 1998) and fetching (Sumbre *et al*. 2005, 2006) movements exhibit stereotypical patterns. The theory that the motor programs are embedded in the low-level controller is also supported by the fact that mechanical or electrical stimulation of amputated octopus arms can elicit extension movements, which are almost kinematically identical to the natural animal behaviors (Sumbre *et al*. 2001). In the soft robots control scheme, the autonomous low-level control programs are deducted by observing the octopus arm movements.

Motivated by the distinct underlying mechanisms for different octopus movements, both open loop and closed loop control are used in the soft robots behavior control architecture. The peak velocities of the octopus reaching movement can be predicted by the initial level of the muscle activation implying an open loop control (Gutfreund *el al*. 2006), while the fetching movement may need the integration of motor and sensory information at both central and peripheral levels suggesting a closed loop control (Sumbre *et al*. 2006). In the open loop control, once a motion is initiated by the high-level controller, the low-level controller carries it out by underlying motor programs without the supervision of the high-level. Therefore, the timing to initiate the motion is essential to harness the complex dynamics, as will be shown in the next subsection.

Master complex soft body dynamics by timing

The high-level controller needs to harness the dynamics of soft robots by specific control parameters. Yekutieli *et al*. (2005b) proposed that the octopus arm extension movement can be fully specified by the activation signal amplitude and the activation travel time. Since their study focused on arm extension, instead of the entire reaching movement, arm base rotation was not considered. We propose that timing (the time to initiate extension movements from the CNS) is also a key parameter to harness the complex arm dynamics in the octopus reaching movement.

The complex body dynamics of soft bodies and structures results in a time delay to transfer the applied forces from the proximal to the distal part. This effect is shown in the octopus by the averaged EMG waveform and the bending time in the arm extension movement that shows an excitation-contraction delay (Gutfreund 1998). The EMG activity precedes the bend propagation in a, to a certain extent, constant manner. Meanwhile, because the CNS (high-level controller) controls the soft arm in open loop for reaching, the timing to initiate movements is fundamental to harness complex dynamics and to enhance the animal's fitness. Here, we assume that the reaching movement can be divided into two phases – arm initiation and arm extension. To the best of our knowledge, there is no biological study concerning the initiation of the octo-

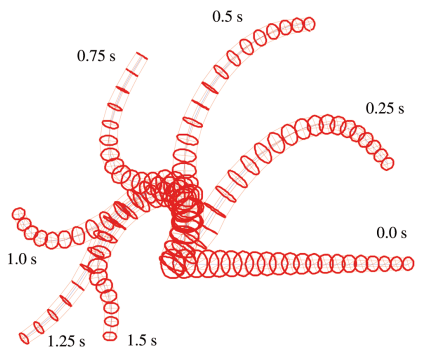


Fig. 2. – The initiation of octopus reaching movement. To facilitate robotic implementation, it is assumed that the octopus needs to initiate its arm by rising up and bringing back its arm before a reaching movement. The initiation sequence is shown by the time stamps.

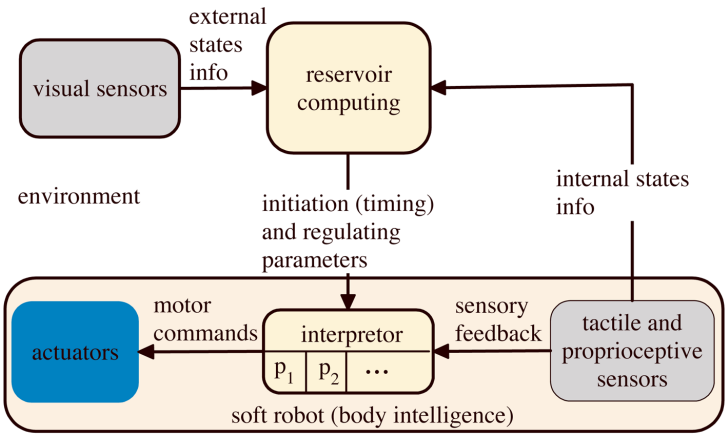


Fig. 3. – Control architecture for soft robots. It contains a dynamical systems based high-level controller (implemented by reservoir computing), a low-level controller which consists of a high-level parameters interpreter and motor programs (P_1, P_2…), a soft robot, and a sensory system. The hierarchical construction simplifies the control of virtually infinite DoFs. The high-level controller learns the arm dynamics and masters it by global parameters, specifically timing. Note that only one arm is shown for clarity, the other arms, if present, are controlled similarly.

pus reaching movement. To facilitate engineering implementation, we assume that the octopus initiates a reaching movement by raising and bringing back its arm, as shown in Fig. 2. The time to launch the second phase (arm extension) is critical. If it is started too early, the initiation of arm has not been fully achieved, for instance, $t = 0.5s$ in Fig. 2, thus it would be hard to achieve successful reaching. However, if it is started too late, the prey may have moved away, the reaching success rate also will not be optimal. By properly adjusting the timing, the octopus can enhance its fitness and the soft robots controller architecture can harness the complex soft body dynamics.

Embed timing into soft robots controller

In order to apply proper timing and achieve appropriate behavior control, the soft robots controller must learn the body dynamics. Studies on some animals show that they learn and represent their body parts by body schema (Hoffman *et al*. 2010). To the best of our knowledge, there is no literature regarding the body schema formation and plasticity of the octopus. In robotics, a body schema can be either explicit or implicit. Because of the complex dynamics of soft robots, which contains many unknown factors, and difficulty to get a precise explicit analytical form representation, an implicit body schema is suitable and biologically plausible.

Timing, as well as the implicit body schema, can be embedded in dynamical systems, which are usually implemented by recurrent neural networks (RNNs) in engineering. The RNNs need to be trained to embed the soft robots body dynamics and proper timing. Traditional RNN training methods use gradient descent techniques on an error surface, but these methods are subject to local minima, slow convergence, and other limitations (Jaeger 2002). Reservoir computing approach provides a new construction and supervised learning principle for RNNs, where a RNN with random internal connection weights is generated as the dynamical reservoir (DR) and only the connection weights from the DR to the readout units have to be adapted in learning (Lukosevicius & Jaeger 2009). This paradigm makes it possible to use any of the many fast linear regression algorithms in RNN training and practical to use RNNs for engineering applications.

Furthermore, our simulation study showed that the high-level controller can use limited information, arm base angle and tactile sensing, from the low-level controller to estimate its performance and gradually learn the robot's dynamics (Nakajima *et al*. 2011a, 2011b). This result supports the idea that the octopus peripheral nervous system can provide limited proprioceptive information in decision-making (Gutnick *et al*. 2011b).

Synthesis of the control architecture

In robotics, reducing control complexity by properly selecting construction materials and smart structure design is termed morphological computation (Paul *et al*. 2006, Pfeifer & Gómez 2009). Exploiting morphological computation can reduce the control complexity of soft robots, thus this factor is an implicit ingredient of the control architecture. By integrating the octopus inspired solutions and embodiment, Fig. 3 shows a schematic diagram of the proposed behavior control architecture for soft robots.

CONCLUSION AND DISCUSSION

The main result of this study – from control challenges, through biological inspirations from the octopus, to their engineering interpretation – is summarized in Table I. The implementation and evaluation of the control architecture proposed in this study for reaching tasks

Table I. – Summary of the soft robots control challenges, from biological inspiration to engineering solutions, and biological hypotheses from the engineering viewpoint

Control challenges	Octopus biological study	Engineering interpretation	Hypotheses from engineering
Control unlimited DoFs	Stereotypical reaching and fetching movements imply embedded motor programs in the peripheral level.	Distribution of the control task between the high-level and the low-level controllers.	There may exist a division of functionality between the CNS and PNS.
	The velocity of the reaching movement can be decided at the beginning by actuation amplitude (Gutfreund 1998).	Open loop control for reaching - once initiated cannot be changed during the execution process.	The timing to initiate reaching movement is important.
	Reaching and fetching might have distinct underlying control mechanisms (Gutfreund *et al.* 1996, Gutfreund 1998, Sumbre *et al.* 2005, 2006).	Use both open loop and closed loop control.	N/A
Control complex body dynamics	Arm extension movement can be fully specified by the activation signal amplitude and travel time (Yekutieli *et al.* 2005b).	The high-level controller sends reaching motion amplitude and speed to the low-level.	The timing to initiate the arm extension is another determining factor of the octopus reaching movement.
	The peripheral system can provide limited proprioceptive information that can be used in decision-making (Gutnick *et al.* 2011b)	The high-level controller can use limited arm states information to learn the robot dynamics.	N/A

has been conducted in simulation studies (Nakajima *et al*. 2011a, 2011b). The results support the idea that the key factor in harnessing a soft arm is to control all the muscles together by initiating the activation with appropriate timing instead of manipulating individual muscles. Further study showed that the timing can be flexibly controlled and autonomously regulated by an echo state network (Kuwabara *et al*. unpubl data). Branson *et al*. (unpubl data) compared the motion of a soft robot simulator controlled by the proposed control architecture with biological data and showed that similar movement trajectories can be generated.

Biologists have conducted a fair amount of studies on the octopus anatomy, behavior, and neurophysiology. These results are critical to soft robotics research, as shown in this paper. However, there are situations in which existing biological studies cannot be adopted by engineers yet information needed is still not available. One such example concerns the octopus sensory system. Significant research concerning the octopus sensory system has been conducted (Young 1971, Wells 1960, 1964; 1978, Byrne *et al*. 2006, Gutfreund *et al*. 2006, Gutnick *et al*. 2011a, 2011b). However, sensory-motor coordination has not been a main focus. This is unfortunate from an engineering point of view as the latter is of particular interest. This situation may be a result of different research motivation and could be resolved by frequent communication and cooperation between biology and engineering communities.

From an engineering perspective, there are several topics or hypotheses that could be further investigated by biologists. To begin with, recent octopus biological research focuses on a limited set of behaviors, especially reaching and fetching. However, building an autonomous robot requires more complete information (Webb 2000); it would be beneficial to study more behaviors in detail to fill these gaps. Next, it is still unclear how the animal deals with the complex dynamics that result from the interaction between the soft body and the underwater environment. It would be beneficial to investigate the body schema of the octopus, for instance, the way that the animal distinguishes its body from the surrounding environment and represents the position and dynamics of its body parts. Moreover, we proposed that timing is critical in controlling the complex nonlinear dynamics. Probably the partial proprioception from the octopus PNS (Wells 1960, Gutnick *et al*. 2011b) can be used to learn and represent the dynamics by the CNS to determine the timing. Further biological study will be valuable to test this timing-based control strategy. Also, multi-arm coordination is an important research topic in robotics. The octopus could provide inspiring solutions because of its ability to control and coordinate its eight arms. Lastly, previous studies showed that the octopus uses many different modes of locomotion, for example, crawling and walking (Mather 1998, Huffard 2006). However, the transition mechanism is still unknown and thus could be a future study. Our study on a physical soft robot platform showed that behavior switching can be achieved by an external input (Li *et al*. unpubl data).

As a concluding remark, biologists and roboticists should treat each other's results with both respect and suspect. Both sides could benefit from the cross-disciplinary cooperation. However, nature does not create optimal solutions even when it provides us with inspiring answers to engineering problems. A working bio-inspired design may not necessarily capture the essences of the biological mechanism.

ACKNOWLEDGEMENTS. – This work is supported by the European Commission in the ICT-FET OCTOPUS Integrating Project (EU project FP7-231608).

REFERENCES

Bekey GA 2005. Autonomous robots: from biological inspiration to implementation and control. Cambridge, Mass.: MIT Press.

Brooks RA 1991a. Intelligence without representation. *Artif Intell* 47(1-3): 139-159.

Brooks RA. 1991b. New approaches to robotics. *Science* 253(5025): 1227-1232.

Byrne RA, Kuba MJ, Meisel DV, Griebel U, Mather JA 2006. Octopus arm choice is strongly influenced by eye use. *Behav Brain Res* 172(2): 195-201.

Cabelguen JM, Bourcier-Lucas C, Dubuc R 2003. Bimodal locomotion elicited by electrical stimulation of the midbrain in the salamander *Notophthalmus viridescens*. *J Neurosci* 23(6): 2434-2439.

Delcomyn F 2004. Insect walking and robotics. *Annu Rev Entomol* 49(1): 51-70.

Edelman GM 2007. Learning in and from brain-based devices. *Science* 318(5853): 1103-1105.

Flash T, Hochner B 2005. Motor primitives in vertebrates and invertebrates. *Curr Opin Neurobiol* 15(6): 660-666.

Gutfreund Y, Flash T, Yarom Y, Fiorito G, Segev I, Hochner B 1996. Organization of octopus arm movements: a model system for studying the control of flexible arms. *J Neurosci* 16(22): 7297-7307.

Gutfreund Y 1998. Patterns of arm muscle activation involved in octopus reaching movements. *J Neurosci* 18(15): 5976-5987.

Gutfreund Y, Matzner H, Flash T, Hochner B 2006. Patterns of motor activity in the isolated nerve cord of the octopus arm. *Biol Bull* 211(3): 212-222.

Gutnick T, Byrne Ruth A, Hochner B, Kuba M 2011a. *Octopus vulgaris* uses visual information to determine the location of its arm. *Curr Biol* 21(6): 460-462.

Gutnick T, Thonhauser K, Zullo L, Hochner B, Kuba MJ 2011b. The Israel Society for Neuroscience 19th Annual Meeting, Eilat, Dec. 12-14, 2010. www.isfn.org.il. *J Mol Neurosci* 45 (Suppl 1): S49.

Hochner B, Shomrat T, Fiorito G 2006. The octopus: a model for a comparative analysis of the evolution of learning and memory mechanisms. *Biol Bull* 210(3): 308-317.

Huffard CL 2006 Locomotion by *Abdopus aculeatus* (Cephalopoda: Octopodidae): walking the line between primary and secondary defenses. *J Exp Biol* 209: 3697-3707.

Hoffmann M, Marques H, Arieta A, Sumioka H, Lungarella M, Pfeifer R 2010. Body schema in robotics: a review. *IEEE Trans Auton Ment Dev* 2(4):304-324.

Ijspeert AJ, Crespi A, Ryczko D, Cabelguen J M 2007. From swimming to walking with a salamander robot driven by a spinal cord model. *Science* 315(5817): 1416-1420.

Jaeger H 2002. Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach. *German Nation Res Center Info Technol Tech Rep* No. 159.

Kang R, Kazakidi A, Guglielmino E, Branson DT, Tsakiris DP, Ekaterinaris JA, Caldwell DG 2011. Dynamic model of a hyperredundant, octopus-like manipulator for underwater applications. *Conf Intelligent Robots Systems (IROS), 2011 IEEE/RSJ Int*: 4054-4059.

Lukosevicius M, Jaeger H 2009. Reservoir computing approaches to recurrent neural network training. *Comput Sci Rev* 3(3): 127-149.

MacKerrow PJ 1995. Introduction to robotics. Sydney: Addison-Wesley.

Mather JA 1998. How do octopuses use their arms? *J Comp Psychol* 112(3): 306-316.

Nakajima K, Li T, Kuppuswamy N, Pfeifer R 2011a. Harnessing the dynamics of soft body with "timing": octopus inspired control via recurrent neural networks. *Proc 15th Int Conf Advanced Robotics (ICAR11)*: 277-284.

Nakajima K, Li T, Kuppuswamy N, Pfeifer R. 2011b. How to harness the dynamics of soft body: timing based control of a simulated octopus arm via recurrent neural networks. *Procedia Comput Sci* 7(0): 246-247.

Nakajima K., Li T., Sumioka H., Cianchetti M., and Pfeifer R. 2011c. Information theoretic analysis on a soft robotic arm inspired by the octopus. *Proc 2011 IEEE Int Conf Robotics Biomimetics (IEEE-ROBIO 2011):* 110-117.

Nie X, Song B, Ge Y, Chen W, Weerasooriya T 2009. Dynamic tensile testing of soft materials. *Exp Mech* 49(4): 451-458.

Paul C, Lungarella M, Iida F 2006. Morphology, control and passive dynamics. *Rob Autom Syst* 54(8): 617-618.

Pfeifer R, Scheier C 1999. Understanding intelligence. Cambridge, Mass.: MIT Press.

Pfeifer R, Bongard J 2006. How the body shapes the way we think: a new view of intelligence. Cambridge, Mass.: MIT Press.

Pfeifer R, Lungarella M, Iida F 2007. Self-organization, embodiment, and biologically inspired robotics. *Science* 318(5853): 1088-1093.

Pfeifer R, Gómez G 2009. Morphological computation – connecting brain, body, and environment. *In* Bernhard S, Edgar K, rner, Olaf S, Helge R, Kenji D eds. Creating Brain-Like Intelligence. Springer-Verlag: 66-83.

Sumbre G, Gutfreund Y, Fiorito G, Flash T, Hochner B 2001. Control of octopus arm extension by a peripheral motor program. *Science* 293(5536): 1845-1848.

Sumbre G, Fiorito G, Flash T, Hochner B 2005. Neurobiology: motor control of flexible octopus arms. *Nature* 433(7026): 595-596.

Sumbre G, Fiorito G, Flash T, Hochner B 2006. Octopuses use a human-like strategy to control precise point-to-point arm movements. *Curr Biol* 16(8): 767-772.

Trivedi D, Rahn CD, Kier WM, Walker ID 2008. Soft robotics: biological inspiration, state of the art, and future research. *Appl Bionics Biomech* 5(3): 99-117.

Webb B 2000. What does robotics offer animal behaviour? *Anim Behav* 60(5): 545-558.

Webb B 2001. Can robots make good models of biological behaviour? *Behav Brain Sci* 24(06): 1033-1050.

Webb B 2002. Robots in invertebrate neuroscience. *Nature* 417(6886): 359-363.

Wells MJ 1960. Proprioception and visual discrimination of orientation in octopus. *J Exp Biol* 37(3): 489-499.

Wells MJ 1964. Tactile discrimination of surface curvature and shape by the octopus. *J Exp Biol* 41(2): 433-445.

Wells MJ 1978. Octopus: Physiology and Behaviour of an Advanced Invertebrate. London; New York: Chapman and Hall.

Yekutieli Y, Sagiv-Zohar R, Aharonov R, Engel Y, Hochner B, Flash T 2005a. Dynamic model of the octopus arm. I. Biomechanics of the octopus reaching movement. *J Neurophysiol* 94(2): 1443-1458.

Yekutieli Y, Sagiv-Zohar R, Hochner B, Flash T 2005b. Dynamic model of the octopus arm. II. Control of reaching movements. *J Neurophysiol* 94(2): 1459-1468.

Young JZ 1963. The number and sizes of nerve cells in octopus. *Proc Zool Soc* 140(2):229-254.

Young JZ 1971. The Anatomy of the Nervous System of *Octopus vulgaris*. Oxford: Clarendon Press.

Received July 18, 2011
Accepted December 13, 2011
Associate Editor: S Boletzky

Appendix B

Harnessing the Dynamics of a Soft Body with "Timing": Octopus-Inspired Control via Recurrent Neural Networks

Nakajima, K., Li, T., Kuppuswamy, N., and Pfeifer, R. (2011). Harnessing the Dynamics of a Soft Body with "Timing": Octopus Inspired Control via Recurrent Neural Networks, In *the 15th International Conference on Advanced Robotics*, pp. 277–284.

This is a version that was accepted for publication. Final version of the article can be found at ieeexplore.ieee.org (doi: 10.1109/ICAR.2011.6088590).

Harnessing the Dynamics of a Soft Body with "Timing": Octopus Inspired Control via Recurrent Neural Networks

Kohei Nakajima, Tao Li, Naveen Kuppuswamy, and Rolf Pfeifer

***Abstract*— This study aims to explore a control architecture that enables the control of a soft and flexible octopus-like arm for an object reaching task. Inspired by the division of functionality between the central and peripheral nervous systems of a real octopus, we discuss that the important factor of the control is not to regulate the arm muscles one by one but rather to control them globally with appropriate timing, and we propose an architecture equipped with a recurrent neural network (RNN). By setting the task environment for the reaching behavior, and training the network with an incremental learning strategy, we evaluate whether the network is then able to achieve the reaching behavior or not. As a result, we show that the RNN can successfully achieve the reaching behavior, exploiting the physical dynamics of the arm due to the timing based control.**

I. INTRODUCTION

Octopuses have hyper redundant limbs with a virtually unlimited number of degrees of freedoms (DOFs), and their movements are known to be significantly sophisticated [6]. Thus, it has been an excellent test case to learn how to control a soft and flexible body. It is well known that simplification strategies have evolved to reduce the number of control parameters in the movement of the flexible arms of the octopus. That is, the division of functionality between the central nervous system (CNS) and the peripheral nervous system (PNS). In octopus reaching behavior, it is well studied that the CNS only sends an initiation and regulation command to the PNS, and almost all the required control of arm muscles in the reaching behavior are handled by the PNS [7]. Accordingly, several studies have intensively focused on the role of the PNS in the reaching behavior [8]-[9].

In this paper, however, we aim to focus on not only the role of the PNS but also the coordination between the CNS and the PNS. This raises a new challenge. Because of this unique organization of the control system, and although the CNS is only initiating and regulating the activation of the PNS, we discuss that, for the CNS, when to send the command becomes rather critical. In other words, to achieve the reaching behavior, the CNS should estimate the physical dynamics of the arm and harness them with appropriate timing [1]-[3].

This paper is organized as follows. In the next section, we overview the division of functionality between the CNS and the PNS in octopus reaching behavior, discuss the importance of timing in the control, and propose the control scheme motivated by it. In section III, we introduce an octopus arm physical simulator to test the proposed control scheme and the implementation of the controller by neural networks. In section IV, we analyze how it successfully achieves the reaching behavior, and finally, we discuss the implication of the results and future works.

This work was supported by the OCTOPUS IP, EU Project FP7-231608

K. Nakajima, T. Li, N. Kuppuswamy, and R. Pfeifer are with Artificial Intelligence Laboratory, Department of Informatics, University of Zurich, Andreasstrasse 15 , 8050 Zurich, Switzerland. email: nakajima@ifi.uzh.ch

II. BIOLOGICAL BASIS OF OCTOPUS CONTROL

How does a real octopus control its soft and flexible body? In this section, we focus particularly on the reaching behavior of a biological octopus and explore how such behavior is enabled.

A. Relationship between the central and peripheral nervous systems exemplified in reaching behavior

There is a division of functionality between the CNS and the PNS. A relatively small central brain (about 50 million neurons) controls the large, complex, and highly autonomous PNS of the arms (about 300 million neurons), integrates processed information from the visual system, and then issues commands to lower motor centers controlling the elaborated neuromuscular system of the arms. This division of functionality is the outcome of evolutionary selection in order to survive in the diverse environment in the sea.

A typical example showing the effectiveness of this division of functionality is the reaching behavior. Reaching behavior consists of a bend propagation along the arm toward the tip in a highly stereotypical and invariant way. The bend is always created in the dorsal side of the arm as the ventral side of the arm approaches the object. In [7], to see a division of functionality between the CNS and the PNS in reaching behavior, the researchers severed the connection between the arms and the brain and, by applying an electrical stimulation to the axial nerve cord, observed the resulting behavior. Thus, the researchers showed that arm extensions (reaching) can be evoked in arms whose connection with the brain has been severed, suggesting that a major part of this voluntary movement is controlled by a motor program that is confined to the arm's neuromuscular system. They also showed that the reaching behavior was triggered by the stimulation but was not directly driven by the stimuli. The PNS of the octopus did not just drive local reflexes but controlled complex movements involving the entire arm. Because the evoked extensions in denervated octopus arms were qualitatively and kinematically identical to natural arm extensions, an underlying motor program appears to be embedded in the neuromuscular system of the arm, which does not require continuous central control. This finding is

consistent with the remarkable autonomy of the arm's local reflexes and with the elaborate nervous system in each arm, which is connected to the brain by a relatively small number of nerve fibers.

It was suggested that this division of functionality between the CNS and the PNS, and the use of a bend propagation with a limited number of control parameters, greatly simplify control of reaching behavior in the octopus [7]-[9].

B. Biologically inspired control scheme: timing based control

The purpose of this paper is to explore a control architecture that realizes the reaching behavior inspired by the relationship between the octopus's CNS and PNS. Activities of the CNS and PNS while executing the reaching behavior can be summarized as follows:

1) To start the reaching behavior, the arm needs to be lifted and form a curve on the dorsal side.
2) According to the visual information, the CNS sets an appropriate base angle of the arm and issues the commands to the PNS to initiate the bend propagation.
3) The PNS executes the bend propagation.

Indeed, due to this control scheme, the CNS does not have to control the movement of the muscles one by one, and the PNS mainly drives the behavior. Several studies have intensively focused on the role of the PNS in the bend propagation behavior [8]-[9]. However, because of this division of functionality, several questions have naturally emerged. How does the CNS recognize when to apply the command to the PNS with such limited information about these diverse physical dynamics of the arm? How can the CNS wait for the bend propagation to be completed while the PNS is activated? These questions suggest that we need to consider an additional important factor in the control, which is "timing."

Now, imagine that we are holding a long rope in our hands. We can estimate the weight and length of the rope indirectly by swinging it around instead of measuring it directly. Also, due to some training, we can even throw the rope to a certain target. Here, what we can control is only the holding part of the rope, namely the grip. That is to say, to control the rope, we need to harness the physical dynamics of the rope only with partial information, and know the appropriate timing to flick the rope with our wrist [1], [3]. This example captures some important aspects of the CNS's control of the octopus. In the octopus case, the situation is much more complex, since the octopus has not only the physical dynamics with damping and hydrodynamic effects due to the water environment, but also the specific dynamics of the bend propagation introduced by the PNS. Since the CNS cannot control the muscles directly, a single signal at a specific time will affect the behavior critically. In this paper, we mainly focus on this temporal control aspect.

III. MODEL DESCRIPTION

In this section, we propose a control architecture to achieve the reaching behavior in an octopus arm physical simulator

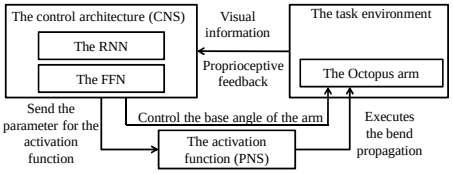


Fig. 1. The control architecture.

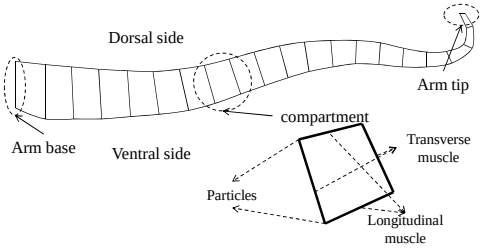


Fig. 2. A schematic diagram showing the simulated octopus arm. Each constant area compartment i is defined by its surrounding longitudinal muscles (ventral side and dorsal side) and transverse muscles. i is the compartment index of the muscle $I = 1, \ldots, N$, where $i = 1$ is the most proximal compartment and N is set to 20 throughout this paper.

following the above explained process. In order to realize the timing based control, the dynamical systems approach will be relevant. An RNN can approximate a nonlinear dynamical system with arbitrary precision. Thus, the control architecture is based on a recurrent neural network (RNN) in combination with a feed forward network (FFN) (Fig. 1). The main body of the RNN controls the angle of the arm base and the timing of the signal sent to the low level control (PNS). The accompanying network decides the power of the signal and the required angle to achieve the reaching behavior.

In the next subsection, we explain the octopus arm model we use in this paper, and then explain our proposed network architecture in detail. Finally, we explain a task and procedure with which to train the network.

A. Octopus arm simulator

The octopus arm model we use in this paper is the one proposed in [4], and is based on the model proposed in [8] with several additional factors. In this subsection, we just discuss the key points of the model.

We use two-dimensional vector particle dynamics for the physical model of the arm. The arm is immersed in water, with external forces considered. It is expressed as a ladder-like structure consisting of point masses and springs (Fig. 2). It is divided into several quadrilateral compartments; the vertices of each compartment are particles, while the edges are massless springs representing muscles. All the entities subject to physics are represented using zero sized particles (point masses), and the motion of each particle is governed by the net force acting on it. There are six forces affecting each particle: gravitational force (F_g), buoyancy force (F_b),

fluid friction force (F_f), inter-particle repulsion force (F_i), pressure force (F_p), and muscle force (F_m). The position of each particle is given by solving the ordinary differential equation, based on these forces at each simulation timestep.

For gravitational force (F_g) and buoyancy force (F_b), we assume that all particles have about the same density as water, so the buoyancy force balances the gravitational force. Fluid friction force (F_f) represents the "resistance" by fluid to particles moving through it. For a particle with velocity v, the force is given by:

$$F_f = -k_f \|v\|^2 \frac{v}{\|v\|}, \tag{1}$$

where k_f is a positive fluid friction constant. Inter-particle repulsion force (F_i) represents the tendency of particles to repel each other. This force is introduced mainly to avoid a bug induced by the collisions of the particles. This force provides a simple but effective substitute for a full solid-body collision model. For a particle at position r, the repulsion force due to another particle at position r' is governed by:

$$F_i = \begin{cases} \frac{k_r}{\|r-r'\|^{p_r}} \frac{r-r'}{\|r-r'\|} & (\|r-r'\| < T_r), \\ 0 & (otherwise), \end{cases} \tag{2}$$

where k_r, p_r, and T_r are constants.

A significant feature of the octopus arm is that its volume is maintained constant: muscular hydrostat [5]-[6]. This feature is expressed in this model as the pressure force (F_p) for maintaining the area of each compartment constant. It works to counteract deviations from the compartment's "desired area" on each of its four particles. At each particle, the force is given by:

$$F_p = k_p(A - A_{des})\hat{n}, \tag{3}$$

where A is the compartment's area, A_{des} is its desired area, and k_p is a constant. $\hat{n}$ is an inward "weighted" vertex normal given by:

$$\hat{n} = \frac{L_1\hat{s}_1 + L_2\hat{s}_2}{\|L_1\hat{s}_1 + L_2\hat{s}_2\|}, \tag{4}$$

where L_1 and L_2 are the lengths of the edges connecting at the particle, and $\hat{s}_1$ and $\hat{s}_2$ are the surface normals of these edges. The presence of the scale factors L_1 and L_2 causes longer edges to receive more pressure than shorter ones.

The octopus arm simulator is regulated by adjusting the muscle forces (F_m) and the base angle of the arms. For muscle forces, we used the linear muscle model described in [4], where the total force of a muscle is composed of a passive force due to the properties of the muscle fibers, and an active force due to the contraction of muscle fibers. The octopus applies an output $a \in [0,1]$ to each of its muscles, which represents the amount of contraction. The effect of a on a muscle is modeled as a force applied to each of the two particles at the ends of the muscle. At each node, this force is given by:

$$F_m = \begin{cases} (a \times F_{ac} + F_{pa})(\frac{L}{L_{rest}} - \overline{L})\hat{m} & (\frac{L}{L_{rest}} > \overline{L}), \\ 0 & (otherwise), \end{cases} \tag{5}$$

where F_{ac} is the muscle's contraction constant, F_{pa} is the reaction force of the muscle fibers, $L(L_t)$ is the muscle's current length, and L_{rest} is the muscle's rest length. $\overline{L}$ is a constant defining the minimum normalized length (the ratio of current to rest length) at which the muscle starts contracting. $\hat{m}$ is an inward-pointing unit vector along the muscle. Note that each particle can be connected to up to three muscles and hence can be subject to up to three different muscle forces. In this paper, the muscles are not controlled individually, but in a group by the activation function. This is an expression of the control of the PNS. The octopus arm starts to move automatically according to the activation function of the muscle with the parameters applied by the CNS. For the activation function, we used the expression introduced in [8]. It is given as follows:

$$a(t,i) = AC_a \cdot \frac{1}{2}\{1 + \tanh[\beta(\frac{t}{\tau} - i + i_0)]\}, \tag{6}$$

where AC_a is the maximum value of the activation parameter and lies between 0 and 1, τ (timestep/segment) is the time for the signal to pass one compartment, i is the compartment index (Fig. 2), and i_0 is the phase-shift parameter that is equal for all compartments and was set at $i_0 = 1$. In addition, τ is set to 100, and the constant β is set to 1. In [8], it was shown that by using the activation function, the bend propagation can be realized, and the behavior of this activation function was investigated in [8]-[9] in detail.

In our model, the arm base is not a spring but is considered to be rigid, and the arm angle can be changed around the center of the base. The arm angle (θ_t (rad)) is controlled, based on the target angle (θ'_{t+1}) of the next timestep and the resting angle (θ_r). In short, θ'_{t+1} controls θ_t of the next timestep directly, and θ_r adjusts the resting angle of θ'_{t+1}. The arm angle (θ_t) is controlled by the target angle (θ'_{t+1}) as follows:

$$\theta_{t+1} = \begin{cases} \theta_t + \gamma & (\theta'_{t+1} - \theta_t > \gamma, 0 < \theta_t < \frac{\pi}{2}) \\ \theta_t - \gamma & (\theta'_{t+1} - \theta_t < -\gamma, 0 < \theta_t < \frac{\pi}{2}) \\ \theta'_{t+1} & (-\gamma \le \theta'_{t+1} - \theta_t \le \gamma, 0 < \theta_t < \frac{\pi}{2}) \\ \theta_{max}(=\frac{\pi}{2}) & (\theta'_{t+1} - \theta_t > 0, \theta_t = \frac{\pi}{2}) \\ \theta_{mini}(=0) & (\theta'_{t+1} - \theta_t < 0, \theta_t = 0) \end{cases} \tag{7}$$

where the arm angle (θ_t) can take a value ranging from θ_{mini} ($=0$) to θ_{max} ($=\frac{\pi}{2}$), and γ is set to 0.01.

We consider here to control the above mentioned arm by neural networks. Our network consists of an RNN and FFN, as mentioned previously. The RNN controls the angle of the next timestep (θ'_{t+1}) and timing; the FFN controls AC_a of the activation function and the resting angle (θ_r). In the next subsection, the network architecture is explained.

B. Network architecture

An important requirement of the control is to set the appropriate arm angle and the parameter for the activation function with the required timing based on partial information about the arm's state. As shown in Fig. 3, two networks are introduced to realize these controls. The first (network A) is an RNN that controls the angle of the base sequentially and decides when to apply the activation function (Fig. 3(a)).

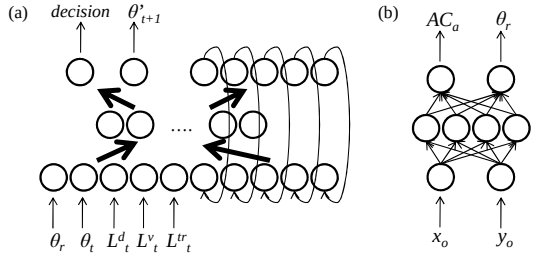


Fig. 3. Network architecture used in this paper. The networks consist of an RNN (network A) (a) and the accompanying FFN (network B) (b). Network A has 10 input neurons (θ_r, θ_t, L^d_t, L^v_t, L^{tr}_t, and 5 recurrent inputs), 20 hidden neurons, and 7 output neurons (decision neuron, θ'_{t+1}, and 5 recurrent outputs). Network B has 2 input neurons (x_o and y_o), 4 hidden neurons, and 2 output neurons (AC_a and θ_r). Note that when the angles are used for the inputs to the network, they are mapped to $(0, \frac{\pi}{2}) \mapsto (0,1)$ linearly and vice versa. According to the value of the decision neuron in network A, it is decided whether the output θ_r of network B is used or not. The network topology we adopted is chosen to be simple but enough to achieve the learning tasks. See the text for detail.

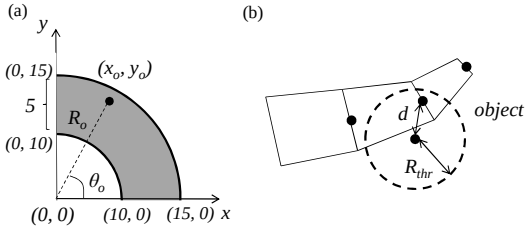


Fig. 4. (a) Task environment. The center of the arm base is set to $(0,0)$. The object is randomly assigned to the shaded region. The position of the object is expressed as $(x_o, y_o) = (R_o \cos\theta_o, R_o \sin\theta_o)$. R_o and θ_o are random real values in the range of $(10.0, 15.0)$ and $(0, \frac{\pi}{2})$, respectively. The octopus never demonstrates the reaching behavior if the target object is either too near or too far away. Accordingly, we set the shaded region to cover compartment numbers 10 to 18, approximately. (b) When we say the object is reached, this means that the distance between the position of the nearest center point of the transverse spring and that of the object is within the threshold distance, R_{thr}. R_{thr} is set to 0.1 in this paper.

The second (network B) is a FFN that regulates the resting angle and AC_a according to the position of the object (Fig. 3(b)). Each network consists of the input layer, the hidden layer, and the output layer, and each neural state is governed by the following equations:

$$\begin{aligned} y_k(t) &= g(net_k(t)), \\ net_k(t) &= \sum_{j=1}^{M} (w_{kj} y_j(t) + bias_k), \\ g(x) &= \frac{1}{1+\exp(-x)}, \end{aligned} \tag{8}$$

where y is the state of the neuron, w_{kj} is the connection weight from the jth neuron to the kth neuron, and *bias* is the bias.

In network A, the inputs are the current arm angle (θ_t), the resting angle (θ_r), and the length of the springs (L^d_t, L^v_t, and L^{tr}_t) of the most proximal compartment; and the outputs are the target angle (θ'_{t+1}) at timestep $t+1$ and the decision neuron to decide to initiate the bend propagation. When the value of the decision neuron is more than 0.5, the bend propagation is started, that is, the output value of network B is adopted for AC_a and θ_r. When the value is less than 0.5, the preparation step for the reaching behavior is started by lifting up the arm. That is, all the springs in the dorsal side of the arm are contracted with $a = 1$ with θ_r. Note that this procedure is set as the default setting and not controlled by network B.

C. Task descriptions and learning

The task and the procedure for training the network are explained in this subsection. The arm should be well controlled by the networks mentioned above and must reach the object assigned in the environment. Initially, the arm is set to the relaxation state with the angle set to 0. The full length of the relaxed arm is 17.3. As revealed in the octopus biology section, the octopus starts to create a curve in the dorsal side of the arm, and through the bend propagation, the arm approaches the object from the ventral side. To realize this situation, the RNN should control the base angle of the arm from 0 to $\frac{\pi}{2}$ and, at the same time, the spring of the dorsal side contracts. The spring of the dorsal side continues to contract, keeping the arm angle to $\frac{\pi}{2}$ and waits for a while until the dorsal side is bent enough, and then, by controlling the base angle and activation function, the arm should reach the object. The object is randomly assigned in the shaded area in Fig. 4(a) when the arm angle gets to $\frac{\pi}{2}$. The arm is defined to reach the object when the distance of the center point of the transverse spring of any compartment of the arm to the object is lower than the R_{thr}(Fig. 4(b)). The important point here is the time lag to wait for the formation of the bend after getting to the angle $\frac{\pi}{2}$. We call this time lag T_{wait}. In the real world, the octopus should control this T_{wait} well and realize the reaching behavior. In this paper, as a first step, we fixed T_{wait} as 2000 (Fig. 5). In the RNN, an output of less than 0.5 for the decision neuron is required, even at the angle of $\frac{\pi}{2}$; and just after passing through the time step of T_{wait}, the decision neuron must bring the output to more than 0.5. During this period, T_{wait}, the several inputs to the RNN are fixed as $\theta_r = 1.0$ and $\theta_t = \frac{\pi}{2}$, so this means that the decision neuron must exceed 0.5 just after passing T_{wait} by understanding the dynamics of the whole arm using only the dynamics of the inputs, L^d_t, L^v_t, and L^{tr}_t. This means it is necessary to design recurrent dynamics. Usually, to handle these time lags, we tend to predefine the time lag as a default setting, or additional stimuli are sent externally to determine the time lag. In this model, we aim to control the time lag autonomously in the network. Once the decision neuron gets to more than 0.5, the extension of the arm starts by controlling the angle of the arm and adjusting the parameters of the activation function. At the same time, the value of θ_r is switched to the output of network B, and parameter AC_a of the activation function is also controlled by the output of network B. When the arm angle gets to the resting angle induced by network B, network A maintains the angle during

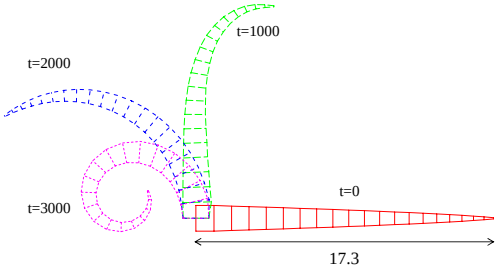


Fig. 5. Transitions of the arm when rotating the arm base with $\theta_{t+1} = \theta_t + \gamma$ while contracting all the springs in the dorsal side with $a = 1$. The cases when $t = 0$, 1000, 2000, and 3000 are shown. Initially, the arm is in the relaxed state with $\theta_t = 0$. Since $\gamma = 0.01$, it takes only about 157 timesteps to get to $\theta_t = \frac{\pi}{2}$. We can see that, to make the curve in the dorsal side, it takes additional timesteps, more than 1000 timesteps, because of the body dynamics in our setting.

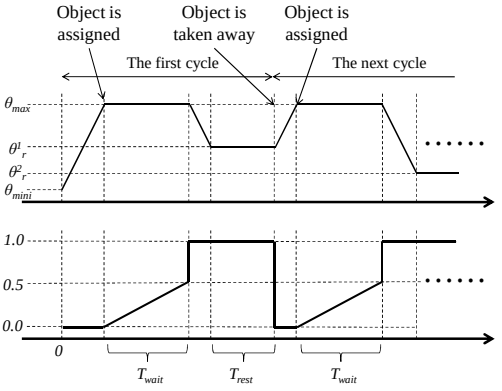


Fig. 6. Target sequence for the RNN (network A). The upper figure shows the sequence for θ'_{t+1}, and the lower figure shows the sequence for the decision neuron. In both figures, the horizontal axes show the timestep. T_{wait} and T_{rest} are set to 2000 and 3000, respectively. θ_{max} and θ_{mini} are $\frac{\pi}{2}$ and 0, respectively. θ^1_r and θ^2_r are the required resting angle of the first cycle, and that of the second cycle, respectively. For θ'_{t+1}, if the object is not in the environment, the arm angle should approach $\frac{\pi}{2}$ with speed γ. However, once the decision neuron exceeds the value 0.5, it should approach θ_r from $\frac{\pi}{2}$ with speed $-\gamma$. For the decision neuron, after the arm angle reaches $\frac{\pi}{2}$, the period of T_{wait} starts. In this period, the value of the decision neuron should approach 0.5 from 0 with speed $\frac{0.5}{T_{wait}}$. See the text for detail.

the T_{rest}. The object is set to disappear from the environment after passing the T_{rest}, and then the value of 1.0 is applied for θ_r. According to this, network A should maintain the value of the decision neuron at less than 0.5 and must control the arm angle until it gets to $\frac{\pi}{2}$. This process is iterated depending on how many cycles we will need to simulate the reaching behavior (Note that the internal dynamics do not need to be adjusted as much as for T_{rest} compared with T_{wait}. Since 1.0 is applied to θ_r when the object disappears from the environment, the disappearance acts as an external signal to tell the time lag.).

To realize the above process, we need to train the net-

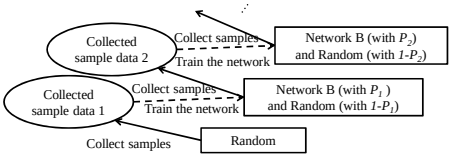


Fig. 7. Network B is trained incrementally.

works. For network A, we designed the target sequence shown in Fig. 6. To train the network, we used a gradient descent method, called back propagation through time (BPTT), based on the output error as follows:

$$\begin{aligned} E_a &= \sum_t \sum_j \tfrac{1}{2}(o^t_j - t^t_j)^2, \\ \Delta w_{ij}(n) &= -\alpha \tfrac{\partial E_a}{\partial w_{ij}} + \beta \Delta w_{ij}(n-1), \end{aligned} \tag{9}$$

where t^t_j, o^t_j, and E_a are the target value of the jth output in timestep t, the state of the jth output neuron, and the error function for network A, respectively. All the connection weights between the input layer and the hidden layer, and between the hidden layer and the output layer are updated. α and β are the learning rate and the momentum rate, respectively. They are set as $(\alpha, \beta) = (10^{-3}, 10^{-3})$. n represents the number of epochs. Each epoch consists of 12000 timesteps, and after each epoch, we conducted BPTT until the 6000th epoch. For the initial connection weights, random real values ranging from $(-1, 1)$ are set. Note that when training network A, the connection to network B is cut off. That is, the output of network B is not used even when the decision neuron gets to more than 0.5. Instead, θ_r and AC_a are applied at random. Note that, although AC_a is not applied to the input of network A, AC_a has an indirect effect on the input of network A via the arm dynamics, L^d_t, L^v_t, and L^{tr}_t. The important thing is that any resting angles with any AC_a must be realized correctly following the target sequence of Fig. 6. That is to say, after learning, network A can control the angle correctly as required, and extend the arm for any outputs coming from network B.

Network B is required to exploit the physical dynamics of the arm induced by network A and successfully achieve the reaching behavior to the object by generating proper activation amplitude AC_a and the resting angle θ_r. Thus, for the training of network B, we use an incremental learning strategy (Fig. 7). It is carried out using network A after network A has finished learning. At first, in network B, the connection weight is assigned at random in the range of $(-1, 1)$ like network A. Using this network, the position of the object assigned randomly in the environment is given as the inputs, (x_o, y_o), and the corresponding outputs, (AC_a, θ_r), are adopted to control the arm with the probability P. If this is not adopted (in probability $1-P$), AC_a and θ_r are chosen at random. By iterating this process, we collect a set of inputs and outputs that successfully reached the object as a target sample. After 100 sets of this target sample are collected, the training of network B is carried out by using the back

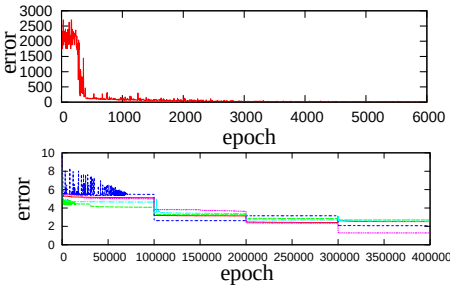


Fig. 8. Plots showing E_a and E_b in each learning epoch. The upper diagram shows the plots of E_a, and the lower shows those of E_b.

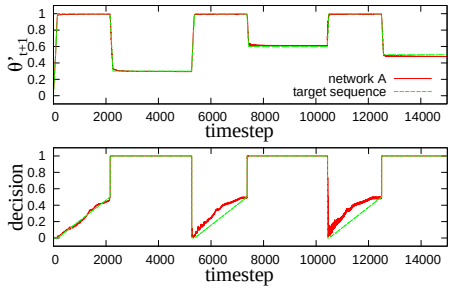


Fig. 10. Examples of the output dynamics of the decision neuron and θ'_{t+1} of the trained network A. For comparison, the target sequence is overlaid for each plot.

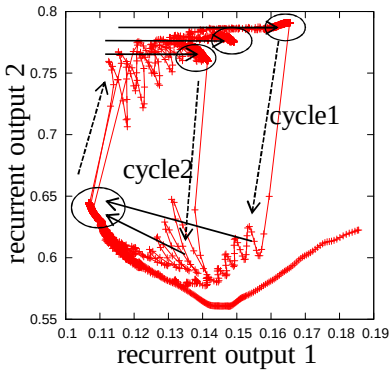


Fig. 11. Corresponding recurrent dynamics for the same simulation in Fig. 10. Two recurrent outputs are used to plot the diagram.

propagation method. It is expressed as follows:

$$E_b = \sum_s \sum_j \frac{1}{2}(o_j^s - t_j^s)^2,$$
$$\Delta w_{ij}(n) = -\alpha' \frac{\partial E_b}{\partial w_{ij}} + \beta' \Delta w_{ij}(n-1), \quad (10)$$

where t_j^s, o_j^s, and E_b are the sth collected target values of the jth output, the state of the jth output neuron to the sth target input, and the error function for network B, respectively. All the connection weights between the input layer and the hidden layer, and between the hidden layer and the output layer are updated. α' and β' are the learning rate and the momentum rate, respectively. They are set as $(\alpha', \beta') = (10^{-2}, 10^{-3})$. n represents the number of epochs. This operation was repeated four times, varying the value of P incrementally as 0, 0.25, 0.5, and 0.75. For each operation, 100 thousand learning epochs were carried out (400 thousand learning epochs in total). This overall procedure was carried out for 5 trials, changing the initial value of the connection weight, and was used for the analysis. It must be noted that this procedure is quite different from taking many successful sets of target samples at once for learning. Taking many target data at once means that all the target data were handled equally. However, in the incremental learning strategy, each set of target data is constrained by the developmental stage of the current network [1].

IV. RESULTS

The results of the training are depicted in Fig. 8. It shows the value of the error function in each learning epoch for networks A and B. For network A, we can see that the error function converged to around 0 rapidly; on the other hand, for network B, the error function never converged to around 0 in all the 5 trials. Fig. 9 shows the successful reaching behavior after learning. We can see that, according to the position of the object, θ_r and AC_a are well controlled to achieve the bend propagation and reach the object.

In this section, by using the trained networks, we will see how the networks achieve the reaching behavior to the object in detail. In the next subsection, we will see how the time lag of T_{wait} is achieved by the network, and analyze the internal dynamics that enable it. Then, we will see how the trained reaching behavior is related to the position of the object in the environment and discuss the relevance of the body dynamics to the control.

A. Analysis of internal dynamics

Fig. 10 shows the dynamics of θ'_{t+1} and the decision neuron. For the decision neuron, although its dynamics are not completely equal to the training sequence, we can observe that it successfully regulates the timing of T_{wait}. By analyzing the corresponding recurrent dynamics, we found that they are switching the point attractors to regulate the dynamics of θ'_{t+1} and the decision neuron (Fig. 11). Especially, T_{wait} is realized by the slow relaxation dynamics to the point attractor (lower left in Fig. 11). Once the dynamics reach this attractor, the decision neuron outputs the value exceeding 0.5, and then the regulation of the θ'_{t+1} starts. Accordingly, the recurrent dynamics jump out from the attractor and aim for another attractor (upper right in Fig. 11) corresponding to the θ_r which is required by network B.

B. Success rate of the reaching behavior

To evaluate the performance of the reaching behavior, we analyzed the success rate. To calculate it, we at first

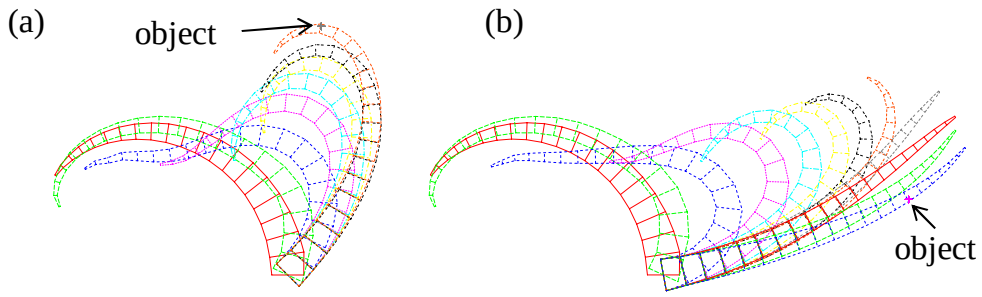


Fig. 9. Examples of the arm dynamics during the reaching behavior. (a) The position of the object was set to $(x_o, y_o) = (1.6, 11.7)$. The output produced by network B was $(AC_a, \theta_r) = (0.51, 0.50)$. Every 200 timesteps from 2100 to 3500 timesteps are plotted. (b) The position of the object was set to $(x_o, y_o) = (11.5, 3.5)$. The output produced by network B was $(AC_a, \theta_r) = (0.74, 0.12)$. Every 200 timesteps from 2100 to 4300 timesteps are plotted.

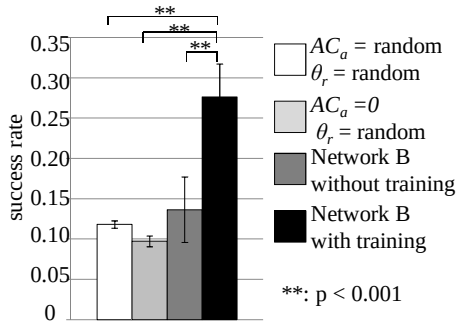


Fig. 12. Comparison of success rates. Success rates between four cases are compared, namely $(AC_a, \theta_r) = (random, random)$, $(AC_a, \theta_r) = (0, random)$, network B without training, and network B with training. The data show the averaged value for 5 trials each and the error bars show the standard deviation. They were $0.12(\pm 0.0045)$, $0.10(\pm 0.0067)$, $0.14(\pm 0.041)$, and $0.28(\pm 0.041)$, respectively. Asterisks indicate significant differences, $**$: $p < 0.001$.

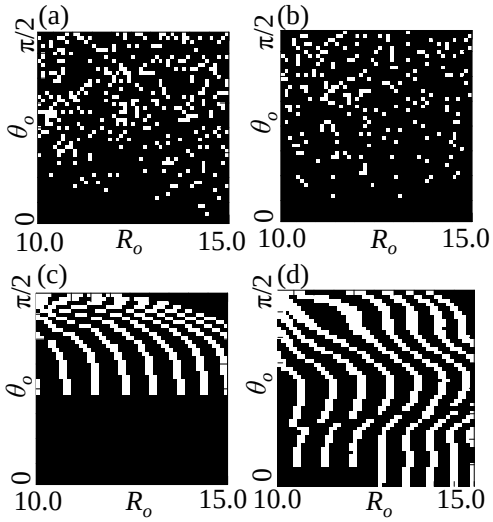


Fig. 13. Comparisons of successful reaching according to the position of the object in the $R_o - \theta_o$ plane. White dots show successes, and black dots show failures. (a) When both AC_a and θ_r are chosen randomly. (b) When $AC_a = 0$ and θ_r are chosen randomly. (c) When network B with random connection weights is used (without training). (d) When network B after training is used.

discretized the $R_o - \theta_o$ plane $((10.0, 15.0) \times (0, \frac{\pi}{2}))$ into 50×50 grids and assigned the object to a grid. We then observed whether the networks could control the arm in order to reach the object. If the arm reached the object, we called the case a success; otherwise it was a failure. By iterating this procedure for all of the grids (2500), we calculated the rate of successful reaching over the $R_o - \theta_o$ plane: $\frac{the\ number\ of\ successes}{the\ number\ of\ all\ the\ grids}$. The result of the calculation is defined as the success rate in this paper. We compared the success rate between four cases. In the first case, for AC_a and θ_r, without using network B, we applied the random real value in the range of $(0, 1)$. For the second case, the application was the same, but for AC_a, we assigned 0. This case is characterized by the control of the arm using only the physical dynamics induced by the control of θ_r. In the third case, we used an untrained network B. And in the fourth case, we used the network B we developed. As a result, we observed a significant difference between the performance of our developed network and other cases (Fig. 12), showing that the reach was significantly improved. The performance was twice as high as the others. Also, in Fig. 13, we can clearly confirm that the reaching is drastically improved compared with the other cases. Moreover, in Fig. 13 (c) and (d), we can find a structure in the $R_o - \theta_o$ plane. This structure is expected to be generated as a consequence of the interaction between the physical dynamics of the arm and the output of the networks. To see how the network deals with the physical dynamics of the arm in the successful reaching, we analyzed the relationships between θ_o and θ_r. If the network is not using the physical dynamics of the arm, θ_r tends to approach θ_o since the relaxation state of the arm

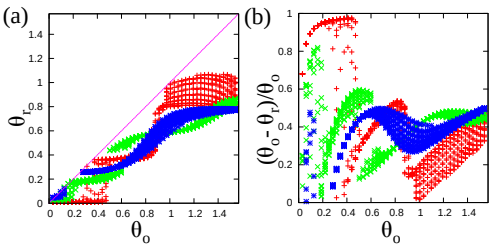


Fig. 14. The relations between θ_o and θ_r. For θ_r, only the data of network B from 3 trials, whose reaching behaviors were successful, are used. (a) $\theta_o - \theta_r$ plot. The line shown in the figure shows $\theta_r = \theta_o$. (b) The values of $\frac{(\theta_o - \theta_r)}{\theta_o}$ according to θ_o are plotted.

is a straight line. If θ_r is less than θ_o, this means that the network is taking the dynamics of the arm into account to reach the object (Fig. 14(a)). To see the effect clearly, we also observed $\frac{(\theta_o - \theta_r)}{\theta_o}$ (Fig. 14(b)). If this value is large, it might be conjectured that the network is taking the dynamics of the arm into account to achieve the successful reaching. Note that these observations are not valid when θ_o is too small, since if θ_o is around 0, it is obvious that there is less space for θ_r to be successfully controlled, and for $\frac{(\theta_o - \theta_r)}{\theta_o}$, the relatively small change in the values of the variables will affect this measure significantly. See Fig. 14. We can thus clearly capture the effect of the physical dynamics on the network. What is interesting is that the relationship between θ_r and θ_o is not linear, which means that, depending on the position (angle) of the object in the environment, the network decides the specific way of controlling the arm in the reaching behavior.

V. CONCLUSIONS AND DISCUSSIONS

In this paper, we proposed a control architecture that was motivated by the division of functionality of the octopus's CNS and PNS in the reaching behavior by using the RNN. By adopting the incremental learning method, we showed that the network can successfully achieve the reaching behavior. Although the performance of the trained networks was significantly higher than the other cases, the absolute value was not as high, namely 0.28 (Fig. 12). What is causing this? One reason would be the setting of $R_{thr} = 0.1$. Considering the ratio between the R_{thr} and the length of the arm at rest, this might be a rather crucial setting. Moreover, there might be regions that are inaccessible in the environment because of the physical constraint in our current setting. Several experiments varying the value of R_{thr} with different environmental settings should be explored in future work.

The important points of our modeling are the timing and the resulting execution of the body dynamics. We showed that the RNN can autonomously realize the time lag based control, and that the accompanying network can take the body dynamics into account in the control of the reaching behavior. In this regard, we fixed the time lag to $T_{wait} = 2000$ as a first step in this paper. For future work, we will explore an architecture that can autonomously control this time lag in order to achieve the reaching behavior.

VI. ACKNOWLEDGMENTS

The authors would like to acknowledge Dr. Michael J. Kuba and Dr. Letizia Zullo for their contribution up to date expertise in octopus biology.

REFERENCES

[1] R. Pfeifer and C. Scheier, *Understanding Intelligence*, Cambridge, MA: MIT Press; 1999.
[2] R. Pfeifer and J. C. Bongard, *How the Body Shapes the Way We Think: A New View of Intelligence*, Cambridge MA; MIT Press, 2006.
[3] R. Pfeifer, M. Lungarella and F. Iida, "Self-Organization, Embodiment, and Biologically Inspired Robotics," *Science*, vol. 318, pp. 1088-1093, 2007.
[4] D. Castonguay and S. Mannor, "Description of the environment's physics," Retrieved July 30, 2010, from McGill University, Centre for Intelligent Machines, website: http://www.cim.mcgill.ca/ dcasto4/octopus/physics.pdf, 2006.
[5] C. Laschi, B. Mazzolai, V. Mattoli, M. Cianchetti and P. Dario, "Design of a biomimetic robotic octopus arm," *Bioinspired Biomimetics*, vol.4, 015006 (p. 8), 2009.
[6] W. M. Kier and K. K. Smith, "Tongues, tentacles and trunks: the biomechanics of movement in muscular-hydrostats," *Zoological Journal of the Linnean Society*, vol. 83, pp. 307-324, 1985.
[7] G. Sumbre, Y. Gutfreund, G. Fiorito, T. Flash and B. Hochner, "Control of Octopus Arm Extension by a Peripheral Motor Program," *Science*, vol. 293, pp. 1845-1848, 2001.
[8] Y. Yekutieli, R. Sagiv-Zohar, R. Aharonov, Y. Engel, B. Hochner and T. Flash, "Dynamic Model of the Octopus Arm. I. Biomechanics of the Octopus Reaching Movement," *J. Neurophysiol.*, vol. 94, pp. 1443-1458, 2005.
[9] Y. Yekutieli, R. Sagiv-Zohar, B. Hochner and T. Flash, "Dynamic Model of the Octopus Arm. II. Control of Reaching Movements," *J. Neurophysiol.*, vol. 94, pp. 1459-1468, 2005.

Appendix C

Behavior Switching by Using Reservoir Computing for a Soft Robotic Arm

Li, T., Nakajima, K., Cianchetti, M., Laschi, C., and Pfeifer, R. (2012). Behavior Switching by Using Reservoir Computing for a Soft Robotic Arm, In *the 2012 IEEE International Conference on Robotics and Automation*, pp. 4918–4924.

This is a version that was accepted for publication. Final version of the article can be found at ieeexplore.ieee.org (doi: 10.1109/ICRA.2012.6225366).

Behavior Switching Using Reservoir Computing for a Soft Robotic Arm

Tao Li, Kohei Nakajima, Matteo Cianchetti, Cecilia Laschi, and Rolf Pfeifer

***Abstract*— Soft robots have significant advantages over traditional robots made of rigid materials. However, controlling this type of robot by conventional approaches is difficult. Reservoir computing has been demonstrated to be an effective approach for achieving rapid learning in benchmark tasks and conventional robots. In this study, we investigated the feasibility and capacity of the reservoir computing approach to embedding and switching between multiple behaviors in a on-line manner in a soft robotic arm. The result shows that this approach can successfully achieve this task.**

I. Introduction

Soft robotics represents one of the new trends and challenges in biologically inspired robots[1]. Traditionally, robots are made of rigid materials and resemble the articulated structure of vertebrates. Motivated by the fact that soft materials are ubiquitous in living creatures, new types of robots that adopt elastic elements in their construction have been developed in recent years [2]. Soft robots have potential advantages over traditional rigid ones in terms of morphological flexibility and interaction safety. Soft robots' flexibility means they could be used as, for example, search and rescue robots, which could crawl through rubble and squeeze into small spaces, and minimal-invasive operation devices. In this study, we take the extreme of softness and consider soft robots whose body or major functioning parts are constructed exclusively of elastic elements. Some encouraging instances of these extremely soft robots have been developed in the last few years [3], [4], [5]. However, their functions are mainly achieved by smart structure design while the controllers are either not addressed or use traditional methods. There are enormous challenges in controlling soft robots since soft materials exhibit highly complex and time-varying dynamics under actuation and the expansion of the applied forces is hard to predict in the structure [6], just to name several. Facing these difficulties, one could envision that traditional robot control methods based on rigid kinematics and dynamics are hard to apply to soft robots in general. There is still no efficient method tailored for controlling soft robots. In this study, we focus on achieving behavior switching of soft robots.

The octopus is a good source of inspiration for learning a control strategy for soft robots. As an inspiration for soft robot construction, octopus arms are extremely compliant and exhibit complex dynamics. However, the octopus controls its soft arms flexibly and precisely to perform various behaviors [7], such as reaching [8], [9] for an object, catching it, and bringing it [10] to its mouth in a varying and often uncertain environment. Our previous study on an octopus-inspired soft robots control scheme proposed that timing, the time to initial motions, is an important factor in controlling soft robots [11], [12], [13], [14], [15]. To implement timing in a robot controller, recurrent neural networks (RNNs) are normally used. However, traditional supervised training methods for RNNs use gradient descent techniques and are subject to local minima, slow convergence, instability, and other limitations [16]. Reservoir computing [16], [17], [18] is a new way to construct and train the RNNs. In this approach, only the connection weights from the reservoir to the output nodes are trained; thus, many fast linear regression algorithms can be used. This characteristic makes it realistic to use reservoir computing in physical robotic platforms.

The main work of this study is as follows: (1) evaluate reservoir computing approach by using different type of sensors with distinct noise characteristic; (2) demonstrate that reservoir computing approach can be used to embed and switch among multiple sequential behaviors in a physical soft robotic platform; (3) and analyze the stability of reservoir to sensor noises. In this paper, we used the reservoir architecture called echo state network (ESN).

II. Experiment Setting

An experimental platform equipped with a soft robotic arm was built to evaluate the interaction among the controller, the soft body, and the environment. The platform setup, data acquisition, and experimental procedure are presented in this section.

A. Platform setup

The platform setup is shown in Fig. 1(a). It consists of a soft robotic arm, its actuation, sensing, control systems, and a water tank containing fresh water as the underwater environment. The soft robotic arm, which mimics the morphology of an octopus arm, is based on the prototype proposed in [19]. It is made of commercially available silicone rubber (ECOFLEX^{TM} 00-30), which has similar density and Young's modulus as the octopus arm [19]. The total length of the cone-shaped soft arm is 310 mm, with an actuated part of 80 mm, measured from the base. The rest 230 mm is passively driven. The actuated part has two nonextensible fishing cables embedded symmetrically to the center of the arm, as shown by the dashed lines in Fig. 1(b). Using two servo motors (Dynamixel^{TM} AX-12A+), the soft

* This work is supported by the European Commission in the ICT-FET OCTOPUS Integrating Project (EU project FP7-231608).

T. Li, K. Nakajima, and R. Pfeifer are with the Artificial Intelligence Laboratory, Department of Informatics, University of Zurich, 8050 Zurich, Switzerland, email: taoli@ifi.uzh.ch

M. Cianchetti and C. Laschi are with the Advanced Robotics Technology and Systems Laboratory, Scuola Superiore Sant'Anna, 56100 Pisa, Italy.

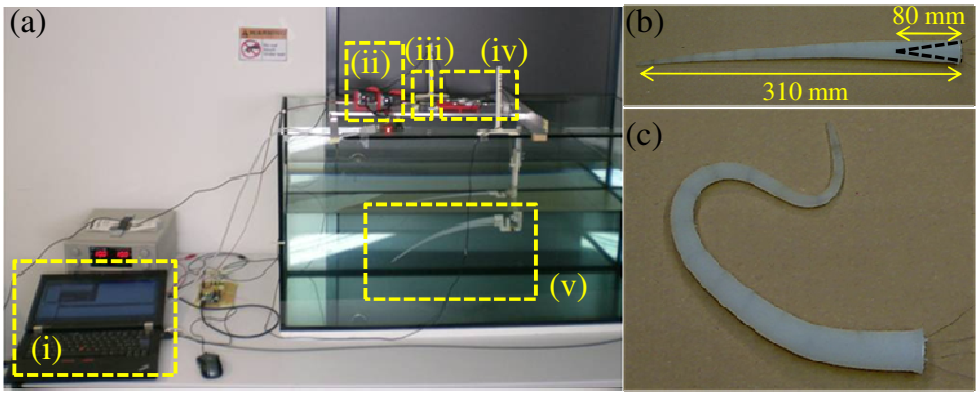


Fig. 1. (a) The experimental platform. It consists of a laptop PC (i), two servo motors (ii), one camera (iii), two force sensors (iv), and a soft robotic arm (v). (b) The soft robotic arm used in this paper. Dashed lines represent the cables embedded in the arm. (c) The robotic arm is made of silicone rubber and thus can be bent at any point and to any direction.

arm is driven by pulling the two cables embedded in the actuated part of the arm. The cable tensions are measured by two force sensors (KD24S from ME-Meβ system GmbH). The force sensor signals are amplified and sent to a PC serial port through an Arduino^{TM} UNO board, whose ADC outputs integer values between 0 and 1023, which correspond linearly to forces of 0 to 10 N. Thus, the unit of force sensor data is about 0.01 N. The force unit is designated by [POS] in this paper for clarity,. The servo motor positions are also sent to the PC as sensory inputs by integer values from 0 to 1023, which correspond linearly to angles of 0 to 300 degrees. The unit of position sensors is designated by [FCE], which is about 0.29 degrees. A camera (Logitech^{TM} Webcame Pro 9000) is placed on the top of the platform to record the the soft silicone arm motion.

A *Java* program running on a laptop PC receives the sensor signals explained above and sends out the motor commands to the servo motors. The unit of timestep, in this paper, is one sensing and actuation loop of the control program. Reservoir is prepared in the program to generate sensory-motor loop. For the sensor signals (S), it allows to take either the force sensor readings or the servo motor positions. The sensor selection depends on experimental setting explained below. We adopt three variables for the motor commands, which are the moving direction for each motor and their common speed. The overall settings of the reservoir and experimental procedures are explained in detail in the following sections.

The servo motor position, designated by integers between 0 and 1023, as described above, is adjusted according to the motor command (direction) and speed. The servo motor speed (v) is the motor position change per timestep and thus has the unit of [POS/t]. It can be set from 10 [POS/t] to 40 [POS/t] considering the limitation of the platform - a speed slower than 10 [POS/t] cannot exhibit the dynamics of the soft arm, while a speed faster than 40 [POS/t] would cause the servo motor to overheat. In this paper, motor commands are set as binary values, $M = \{+1, -1\}$. If the command gives $+1$ or -1, the motor is controlled to move from the current position toward the maximum position (L_{max}) or the relaxed position (L_{relax}) with the speed v, respectively. For each motor, L_{max} was determined so as not to cause the tip of the arm to touch the walls of the water tank during its movement. Note that the motor command does not always take the roller position to L_{max} or L_{relax}, but rather decides the motor moving direction for each timestep. Also, if the command gives $+1$ or -1 when the current position is in L_{max} or L_{relax}, respectively, then the position will stay unchanged for one timestep.

B. Reservoir Computing

1) Sensory - motor mapping: As we explained above, we prepared two types of sensors for the reservoir. The first type are force sensors (S_f) and the second are position sensors (S_p). Since the sensors measures two motors (cables), we describe them as $S1$ (S_f1 or S_p1) and $S2$ (S_f2 or S_p2). Meanwhile, since we aim to embed multiple behaviors into the network, we adopt control signals (C) as a input to the reservoir. Here, C is defined as a random real value in [0.0 1.0]. For example, if we want to control three behaviors, then we divide the range [0.0 1.0] equally and assign a control signal as (0.33, 0.66, 1.0) for each behavior. The correspondence between the assigned values and the behaviors is randomly determined and fixed in the training phase, as explained below. As a summary, we have $S1$, $S2$, and C as the input to the reservoir (Fig.2). For the outputs of the reservoir, we adopt previous explained motor commands (M) and speed (v). Since the two cables are controlled independently, there are two motor commands - $M1$ and $M2$. As a summary, there are three reservoir outputs in total (Fig.2).

2) Network architecture - Echo State Network (ESN): Fig.2 shows the ESN used in this study. It consists of four

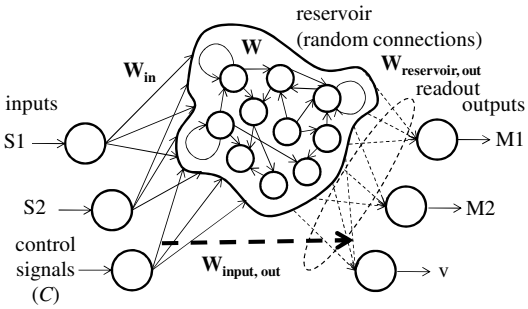


Fig. 2. Network architecture used in this paper. There are three input nodes (sensory inputs (*S*1, *S*2), and a control signal (*C*)), 200 reservoir neurons, and three output nodes (motor rotation directions (*M*1, *M*2), and motor speed (*v*)). All the connections are fully connected but not shown for simplicity. See text for details.

types of connection weights. The first type is connection weights from input nodes to the reservoir neurons (W_{in}, size 200×3). The second is connection weights that connect the reservoir neurons to each other (W, size 200×200). The third are direct connection from the input nodes to the output nodes ($W_{input,out}$, size 3×3). The last are connection weights from the reservoir neurons to the output nodes ($W_{reservoir,out}$, size 3×200). We use 200 neurons to construct the reservoir throughout this paper. Connection weights W_{in} are random real value from [0.0 1.0] and fixed throughout all the experiments. For the setting of W, we used the method introduced in [16], which can be summarized as the following procedure: (a). Randomly generate an internal weight matrix (W'). (b). Normalize W' to a matrix W'' with unit spectral radius by applying $W'' = W'/|\lambda_{max}|$, where $|\lambda_{max}|$ is the spectral radius of W'. (c) Scale W'' to $W = \alpha W''$, where $\alpha < 1$, whereby W has a spectral radius of α. α is set to 0.9, which is determined heuristically in this study. $W_{reservoir,out}$ and $W_{input,out}$ are the weights that will be adapted in the training phase. W_{out} is used to represent the concatenation of the two output weights matrix, $W_{reservoir,out}$ and $W_{input,out}$. $W_{out} := W_{reservoir,out} \oplus W_{input,out}$. We define input sequence in n timesteps as $u(n) = (u_1(n), u_2(n), u_3(n))$, where $u_1(n) = S1(n)$, $u_2(n) = S2(n)$, and $u_3(n) = C(n)$. The state of the neurons of the reservoir is $x(n) = (x_1(n), x_2(n), ..., x_{200}(n))$. The concatenation of the input sequence and neuron states is represented by $o(n) := x(n) \oplus u(n)$. The output of the network is $y(n) = (y_1(n), y_2(n), y_3(n))$, where $y_1(n) = M1(n)$, $y_2(n) = M2(n)$, and $y_3(n) = v(n)$. Then, the updating rules of the connection weights are defined as:

$$x^T(n+1) = g(W_{in} * u^T(n+1) + W * x^T(n)), \quad (1)$$

$$y^T(n+1) = W_{reservoir,out} * x^T(n+1) + W_{input,out} * u^T(n+1) \quad (2)$$

$$= W_{out} * o^T(n+1), \quad (3)$$

$$D = \Sigma_{t=1}^{T} \Sigma_{i=1}^{N} d_t^i, \quad (4)$$

$$g(x) = \tanh(x). \quad (5)$$

(Note that, we actually applied $g(x)$ to calculate $y(n+1)$ because they are binary values.). Next, in order to determine W_{out}, we need to decide a teacher output ($d(n) = (d_1(n), d_2(n), d_3(n))$). As we will explain in the following section, $d(n)$ is determined according to behaviors we require. The internal states, $o(n)$ for $n = t_{start}$, $t_{start} + 1$, ..., $t_{start} + t_{train}$ are collected into the rows of a state-collecting matrix X of size $t_{train} \times (200+3)$, where t_{start} is the timestep to start collecting the data and t_{train} are the timesteps used for the training data. At the same time, the teacher outputs $d(n)$ are collected into the rows of a matrix T of size $t_{train} \times 3$. Thus, the desired weights are directly obtained by multiplying the pseudoinverse of X (X^*) with T:

$$W_{out} = X^* T. \quad (6)$$

3) Experiment procedure and network training: The aim of this study is to evaluate whether the reservoir computing approach can be reliably applied to the physical soft robotic platform. When we try to embed control inspired by the octopus, we have to embed and combine various types of sequential control. As a preliminary exploration, we design the motor outputs for simple oscillatory behaviors of the robotic arm. By regulating the control signal, we aim to switch those behaviors in an on-line manner. Furthermore, we aim to explore the relation of the types of sensor (that is, the precise position sensors and noisy force sensors) to the reservoir performance. Since the position sensors take the values of servo motor position, they will not be affected by the dynamics of the soft robotic arm. However, the force sensors are expected to be strongly affected by the body dynamics of the soft arm, since they detect the forces on the cables embedded in the arm. We also try to evaluate the robustness to the noise of the reservoir.

The oscillatory behavior of the arm is achieved by alternatively adjusting the forces on the two cables embedded in the soft robotic arm. Initially, one cable (cable 1) is in its relaxed state and the other (cable 2) is in its maximum. Then, the motor driving cable 1 starts increasing the tension on the cable at a constant speed until the cable reaches the predefined maximum position, while cable 2 is driven moving toward the relaxation position. Then, cable 1, which is at its maximum position, starts to go back to the relaxed position, while cable 2 goes to its predefined maximum position. This alternative adjustment continues until a predefined timestep. There are three behaviors defined by different speeds. The three behaviors used in the experiment have speeds of 10 [POS/t] (behavior C), 18 [POS/t] (behavior B), and 26 [POS/t] (behavior A), corresponding to control signal of 1.0, 0.33, and 0.66, respectively. For each of the three behaviors defined in the experiment, the speed is the same for both cables and both directions.

To achieve the behavior switching, three phases are used: teaching, learning, and evaluating. In the teaching phase, the teaching data to be used to train the reservoir readout is generated for 8000 timesteps. A random control signal is generated at the beginning of every 200 timesteps, and the soft robotic arm oscillates at the corresponding speed.

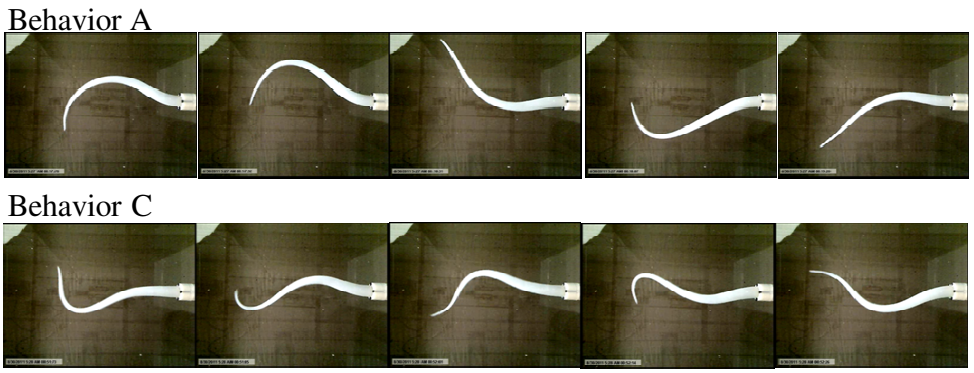


Fig. 3. Typical examples of the soft robotic arm behavior. The upper line shows behavior A, the lower line shows behavior C. Time evolves from left to right. We can see that the amplitude of the oscillatory behavior in behavior A is larger than in behavior C.

We record the control signal and the corresponding motor positions and forces. Then we use the teaching data generated in the teaching phase to train the linear reservoir readout connection weights. The first 200 timesteps teaching data is used to eliminate the effects of the arbitrary starting state and discarded as standard practice. In addition, a random noise is added to the sensor data to enhance the the reservoir's stability. The noise amplitude is determined by considering the sensor output range during the experiment. It is with an amplitude of 26 [POS] for the position sensors and 30 [FCE] for the force sensors. After training, the reservoir is implemented and evaluated to switch among the three behaviors for 5000 timesteps. First, we activate the arm using the same procedure as the teaching phase for 200 timesteps. Then, the reservoir takes a random control signal and generates the corresponding behavior. The generated positions $M(n)$ and the desired positions $d_m(n)$ for both cables at each timestep are recorded.

To evaluate whether the reservoir dynamics is necessary to generate the desired behavior switching, we also tested the setting without reservoir. This is essentially performed by setting the number of reservoir nodes $N = 0$. Therefore, only the connection weight from the input nodes to the output nodes are adapted during the training phase. Further experiments are carried out to analyze the robustness of the reservoir to position sensor noise. Position sensors are used in this experiment. Firstly, a set of training data are collected and used in all the training procedures in this experiment. In the learning phase, we use the same procedure. In the evaluation phase, 10 different levels of noise are added to the position sensor data. The 10 levels of random noise are set from 0 to 90 [POS] (the range of position sensor data is 185 [POS]) with an interval of 10 [POS]. Each noise level is tested 5 times. Error is evaluated by using the root mean square (RMS) of the errors in each timestep: $E_M = \sqrt{\frac{1}{t_{train}} \sum_{n=t_{start}}^{t_{start}+t_{train}} (d_M(n) - M(n))^2}$, $E_v = \sqrt{\frac{1}{t_{train}} \sum_{n=t_{start}}^{t_{start}+t_{train}} (d_v(n) - v(n))^2}$.

III. RESULTS

In this section, the behavior of the soft robotic arm is observed first. Then, the performance of the task to switch among three behaviors by the control signal is evaluated. We compare each case by first adopting the position sensors and then using the force sensors as input. Next, the performance of the controller without reservoir is evaluated to check the importance of the reservoir. Last, the stability of the reservoir is evaluated by adding noise to the position sensor input.

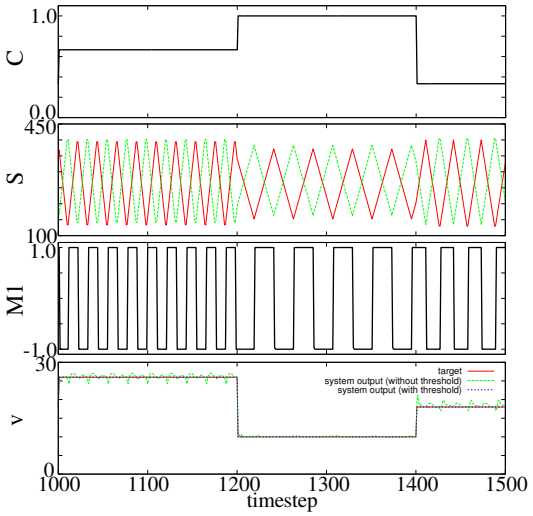


Fig. 5. The plots showing the timesteps from 1000 to 1500 in Fig.4. From the upper line to the lower line, it shows the trajectory of control signal (C), position sensor signal ($S1$)(unit: [POS]), motor command ($M1$), and speed (v)(unit: [POS/t]). In the plots, of the sensor signal, the red line shows $S1$ and the green line shows $S2$.

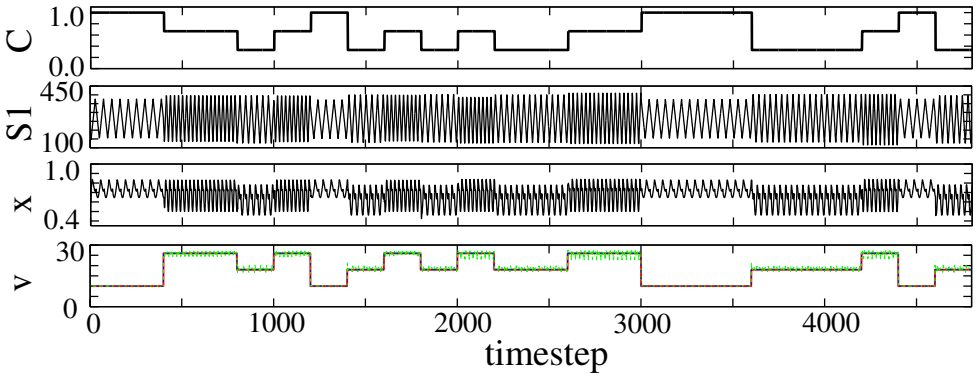


Fig. 4. Examples showing the trajectories of the variables when adopting the position sensors. From the upper to the lower line, it shows the trajectory of the control signal (C), position sensor signal ($S1$)(unit: [POS]), one reservoir neuron (x), and speed (v)(unit: [POS/t]). In the plots showing the trajectory of speed, the red line shows the target trajectory, while the blue line shows the output trajectory. They are overlapped in the plot. Note: units are not shown in the figure due to space limitation

A. Observations

The distinctive feature of a soft arm is the time delay to transmit the motion of the arm generated by the motors from the base to the tip since the arm is soft. This is an intrinsic feature of a soft body that is not observed in a rigid body, for example, a metal stick. As explained in the previous section, the oscillatory behavior is adopted and the speed of the arm oscillation is predefined from large to small in the order of behavior A, B, and C. The diverse behavior of the soft arm, which is controlled by the reservoir computing approach, is shown in Fig.3.

B. The influence of sensor type to reservoir performance

As mentioned in previous sections, the position sensors are not influenced by the diverse behavior of the soft arm as these sensors reflects the value of the angles of the servo motors. On the contrary, the force sensors are venerable to the effect of the dynamics of the soft body as it measures the forces on the two cables.

The behavior switching performance when the position sensors are adopted is shown in Fig.4. It can be seen that the dynamics of sensory data and a reservoir node, as an example, is switched clearly according to the control signal. Furthermore, the switching of the speed is achieved precisely. Fig.5 shows the details between the 1000 and 1500 timesteps. This figure shows that the pattern of the motor command is switched clearly according as the control signal. Next, let us check the case when the force sensors are adopted to reveal the influence of sensor type to reservoir performance. As shown in Fig.6, although the response of the sensors and the dynamics of the reservoir are switched by the control signal, a noisier pattern is observed compared with the case when the position sensors are used. Moreover, some errors can be observed in speed control. Fig.7 shows the details between the timesteps of 2500 and 3000 of Fig.6. Even the frequency of the motor command in each behavior is achieved to some extent, there are clear errors can be seen in the motor commands and motor speed.

C. The necessity of the reservoir

Fig.8 shows the results that no reservoir ($N = 0$) is used. It can be seen that neither the motor speed (v) nor the position sensor data ($S1$) has reached the desired values. The position sensor data ($S1$) is keep the same means motor 1 was not moving. Therefore, the reservoir dynamics is essential to achieve the behavior switching.

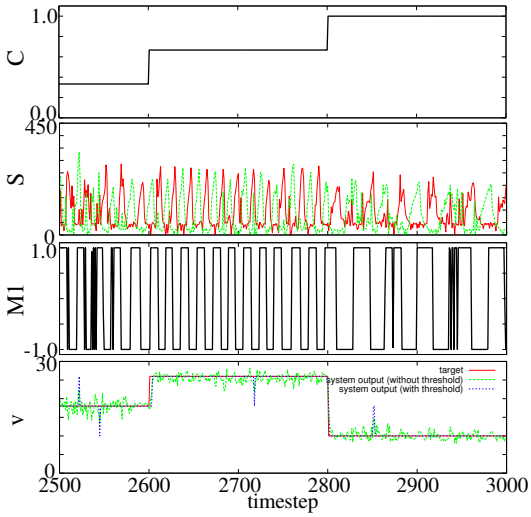


Fig. 7. The plots showing the timesteps from 2500 to 3000 in Fig.6. From the upper line to the lower line, it shows the trajectory of control signal (C), force sensor signal ($S1$)(unit: [FCE]), motor command, and speed (v)(unit: [POS/t]). In the plot of the sensor signal, the red line shows $S1$ and the green line shows $S2$.

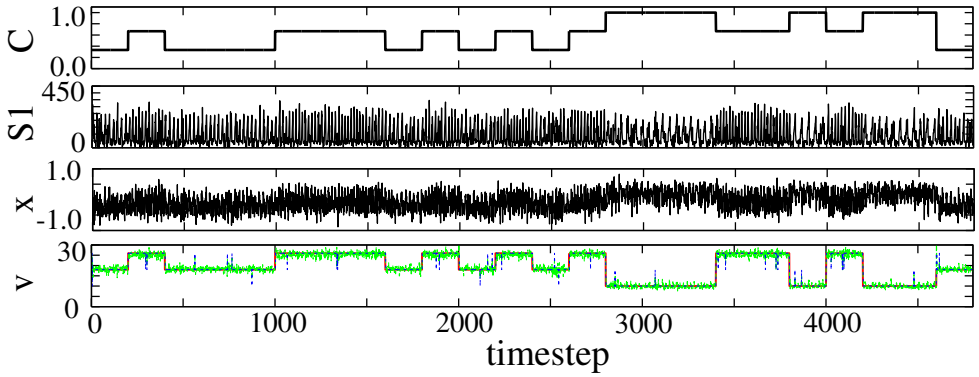


Fig. 6. Examples showing the trajectories of the variables when adopting the force sensors. From the upper to the lower line, it shows the trajectory of the control signal (C), force sensor signal ($S1$)(unit: [FCE]), one reservoir neuron (x), and speed (v)(unit: [POS/t]). In the plots showing the trajectory of speed, the red line shows the target trajectory, while the blue line shows the output trajectory. We can see that the output sometimes shows slightly different value from the target value. Note: units are not shown in the figure due to space limitation

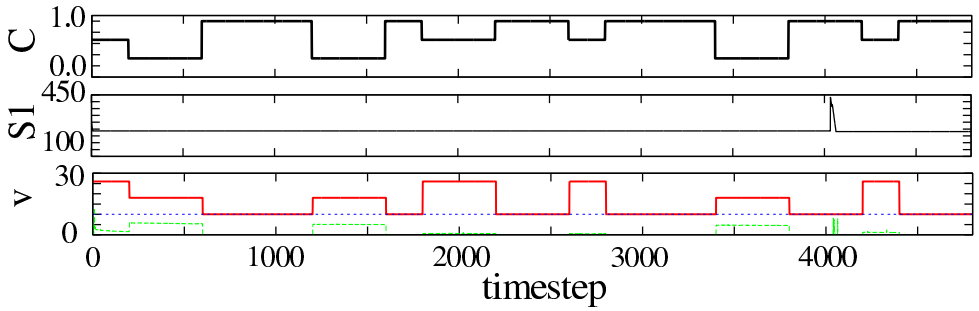


Fig. 8. Examples showing the trajectories of the variables when the reservoir was removed. From the upper to the lower line, it shows the trajectory of the control signal (C), position sensor signal ($S1$)(unit: [POS]), and motor speed (v)(unit: [POS/t]). In the plot showing the trajectory of speed, the red line shows the target trajectory, while the blue line shows the output trajectory. We can see that the output failed to achieve the desired values. Note: units are not shown in the figure due to space limitation

D. Stability to noise

One important criterion in evaluating a physical platforms controller is its robustness to noise. In this section, we estimate the noise impact on the task performance by adding noises to the position sensors data. First, we check the errors when adding different levels of noises to the position sensor data, shown in Fig.9. One can see a trend that both the RMS errors of motor commands and speeds increase with the increasing noise levels. This is consistent with the result shown in previous section that reservoir has better performance when using position sensors, which is not influenced by the soft robot body dynamics and shows less noisy data.

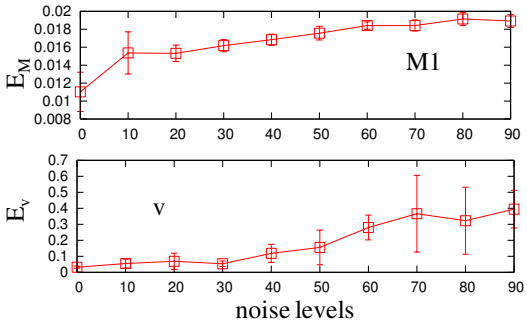


Fig. 9. Plots showing the errors according to the noise levels applied to position sensors. The upper line shows the RMS error plots of $M1$ (E_M), the lower one shows the RMS error of speed v (E_v)

IV. CONCLUSIONS AND DISCUSSIONS

This study shows that it is possible to switch multiple behaviors on-line using reservoir computing in a soft robotic platform. The overall performance was successful to achieve

periodic behaviors. Moreover, when two types of sensors were used (the position sensors and the force sensors), the performances differed. The performance was degraded when the force sensors were used compared with position sensors due to the body dynamics of the soft arm. In addition, when more behaviors were embedded, the performance changed. As the noise to the sensor data increased, the performance gradually degraded, but the performance was not affected so much by increasing the number of behaviors to 9. Finding a way to realize a more stable performance is a future objective.

In reservoir computing, it is possible to realize the simple and robust learning by adjusting only the readout in the training. But it is true that the performance also depends on portions other than the readout, such as the number of nodes and the spectral radius of the reservoir. As a matter of fact, the performance is also changed by the physical platform to be controlled, for instance, the type of sensor, as seen in this study. It is possible to enhance the robustness of the reservoir performance by looking into these issues.

Moreover, two additional aspects can be explored in future work. First, the control signal used to change among the behaviors can be more natural and realistic, for example, using visual sensors and different objects as stimuli. Second, The behavior used in this paper is a very simple periodic behavior. To be notified, the reservoir's performance depends on the task to be realized. In octopus, many interesting behaviors can be observed. For example, in reaching for an object, the octopus uses bending propagation in the arm, and the octopus forms a joint-like structure in the arm when fetching an object. Clearly, further improvements are needed to embed these behaviors. These aspects will be studied in our future work.

References

[1] R. Pfeifer, M. Lungarella, and F. Iida, "Bio-inspired "soft" robotics: The new challenges ahead (periodical style - accepted for publication)," *Communications of the ACM*, to be published.

[2] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, 2008.

[3] E. Steltz, A. Mozeika, N. Rodenberg, E. Brown, and H. M. Jaeger, "Jsel: Jamming skin enabled locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, 2009, pp. 5672–5677.

[4] E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. R. Zakin, H. Lipson, and H. M. Jaeger, "Universal robotic gripper based on the jamming of granular material," *Proceedings of the National Academy of Science*, vol. 107, pp. 18 809–18 814, 2010.

[5] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, 2011.

[6] X. Nie, B. Song, Y. Ge, W. Chen, and T. Weerasooriya, "Dynamic tensile testing of soft materials," *Experimental Machanics*, vol. 49 (4), pp. 451–458, 2009.

[7] T. Gutnick, A. R. Byrne, B. Hochner, and M. Kuba, "Octopus vulgaris uses visual information to determine the location of its arm," *Current biology*, vol. 21 (6), pp. 460–462, 2011.

[8] Y. Gutfreund, T. Flash, Y. Yarom, G. Fiorito, I. Segev, and B. Hochner, "Organization of octopus arm movements: A model system for studying the control of flexible arms," *Journal of Neuroscience*, vol. 16 (22), pp. 7292–7307, 1996.

[9] Y. Gutfreund, "Patterns of arm muscle activation involved in octopus reaching movements," *Journal of Neuroscience*, vol. 18 (15), pp. 5976–5987, 1998.

[10] G. Sumbre, G. Fiorito, T. Flash, and B. Hochner, "Neurobiology: Motor control of flexible octopus arms," *Nature*, vol. 433(7026), pp. 595–596, 2005.

[11] K. Nakajima, T. Li, N. Kuppuswamy, and R. Pfeifer, "Biologically inspired control of a simulated octopus arm via recurrent neural networks," in *Proceedings of the 2011 Genetic and Evolutionary Computation Conference (GECCO 2011)*. ACM press, 2011, pp. 21–22.

[12] ——, "Harnessing the dynamics of a soft body with "timing": Octopus inspired control via recurrent neural networks," in *Proceedings of the 15th IEEE International Conference on Advanced Robotics (ICAR 2011)*, 2011, pp. 277–284.

[13] ——, "How to harness the dynamics of soft body: Timing based control of a simulated octopus arm via recurrent neural networks," *Procedia Computer Science*, vol. 7, pp. 246–247, 2011.

[14] K. Nakajima, T. Li, H. Sumioka, M. Cianchetti, and R. Pfeifer, "Information theoretic analysis on a soft robotic arm inspired by the octopus," in *2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011.

[15] T. Li, K. Nakajima, M. Kuba, T. Gutnick, B. Hochner, and R. Pfeifer, "From the octopus to soft robots control: an octopus inspired behavior control architecture for soft robots (periodical style - accepted for publication)," *Vie et Milieu/ Life and Environment*, to be published.

[16] H. Jaeger, "Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the "echo state network" approach," German National Research Center for Information Technology, Tech. Rep. 159, 2002.

[17] B. Schrauwen, D. Verstraeten, and J. V. Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th European Symposium on Artificial Neural Networks (ESANN2007)*, 2007, pp. 471–482.

[18] W. Maass, T. Natschlaeger, and H. Markram, "Real-time computing without stable states: a new framework for neural computation based on perturbations," *Neural Computation*, vol. 14(11), pp. 2531–2560, 2002.

[19] M. Cianchetti, A. Arienti, M. Follador, B. Mazzolai, P. Dario, and C. Laschi, "Design concept and validation of a robotic arm inspired by the octopus," *Materials Science and Engineering C*, vol. 31, pp. 1230–1239, 2011.

Appendix D

Online Learning Technique for Behavior Switching in a Soft Robotic Arm

Li, T., Nakajima, K., and Pfeifer, R. (2013). Online Learning Technique for Behavior Switching in a Soft Robotic Arm, In *the 2013 IEEE International Conference on Robotics and Automation*, pp. 1288–1294.

Online Learning for Behavior Switching in a Soft Robotic Arm

Tao Li, Kohei Nakajima, and Rolf Pfeifer

Abstract— **Soft robots possess several potential advantages over traditional articulated ones and have attracted significant interest in recent years. However, to control this new type of robots using conventional model-based robotic control approaches is generally ineffective. In this paper, we investigate the challenge to embed and switch among multiple behaviors for an octopus-inspired soft robotic arm. An online learning method for reservoir computing is exploited for this task. This online learning method does not require a separate teaching data collection phase; thus, it has the potential to achieve autonomy in soft robots. Our result shows the feasibility of this approach.**

I. Introduction

Robots constructed of soft materials constitute an emerging research area in robotics [1]–[5]. In terms of morphological flexibility and interaction safeness, soft robots have potential advantages over traditional rigid ones. With virtually unlimited degrees of freedom in their structures, and thus increased flexibility, soft robots have promising application potentials in, for instance, robotic rescue and minimally invasive operations. However, it is challenging to control soft robots due to the fact that soft materials exhibit highly complex and time-varying dynamics when subjected to external forces [6], [7]. The significant deformation of soft robots under normal working conditions violates the basic small strain and linear behavior assumption of engineering mechanics [8]. As such, traditional robot control methods based on the kinematics and dynamics of rigid materials are generally ineffective to apply to soft robots.

The octopus is a living example of an effective control system fully exploiting the benefits of a totally soft body. Unlike many other animals, it has no internal or external skeleton to support its body. However, the octopus is able to control its soft arms dexterously to perform various behaviors [9], such as reaching for an object, manipulating it, and putting it to the mouth in the complex and constantly changing underwater environment [10]–[12]. Studying the control strategy of the octopus by constructing robots with similar morphology and implementing octopus-like behaviors would provide insight into understanding this complex biological control scheme and designing novel control methods for soft robots [13]–[15]. Our previous study on octopus-inspired soft robots proposed that timing (the time to initiate motions from the high-level controller) is an important factor to harness the softness and implement typical octopus behaviors in soft robots [16]–[23]. Dynamical systems are frequently used to embed timing in a controller. As a typical way to implement dynamical systems, recurrent neural networks (RNNs) are conceptually powerful but hard to train. Reservoir computing provides a new way to construct and train the RNNs [24]–[27]. In this approach, only the connection weights from the reservoir (a randomly generated RNN) to the output nodes are trained. Therefore, learning is fast and easy to implement, making it realistic to use reservoir computing in physical robotic platforms. We have used the echo state network (ESN), a pioneering reservoir computing method, to embed and switch among multiple behaviors for a soft robotic arm [28] and to control the moving direction and speed for a multi-arm soft robot [29]. However, these previous studies have two limitations: (1) the control signal to switch behaviors was an arbitrarily defined number in the single-arm study; (2) learning was offline and required a separate teaching data collection phase in both studies.

This study aims to investigate if an online learning method for the ESN could be implemented on an octopus-inspired soft robotic arm to embed and switch among multiple behaviors by visual inputs. Compared with offline batch learning, the online learning method does not require a separate data collection phase. Eliminating the data collection phase allows a robot to develop its skills while interacting with the environment. The online method has the potential to deal with unexpected changes in the robot structure and the working environment. This characteristic is critical to achieve autonomy for robots interacting with the physical environment.

The rest of this paper is organized into three sections. First, the soft robotic experimental platform, the task, the ESN architecture, and its online learning algorithm are described. Subsequently, the performance of the control strategy is analyzed. Finally, we discuss implication and conclude.

II. Experimental Setting

This section presents the experimental platform, task, ESN control architecture, sensorimotor loop, experimental procedure and online learning algorithm.

A. Platform setup

The platform setup is shown in Fig. 1(a). It consists of an octopus-inspired soft robotic arm, its actuation, sensing, and control systems, and a water tank containing fresh water as the underwater experimental environment. The soft robotic arm, which is made of commercially available silicone rubber (SMOOTH-ON ECOFLEX^{TM} 00-30) and based on the arm

This work was supported by the European Commision in the ICT-FET OCTOPUS Project (EU project FP7-231608).

T. Li, K. Nakajima, and R. Pfeifer are with the Artificial Intelligence Laboratory, Department of Informatics, University of Zurich, 8050 Zurich, Switzerland. K. Nakajima is also with the Bio-Inspired Robotics Laboratory, Department of Mechanical and Process Engineering, ETH Zurich, 8092 Zurich, Switzerland. Email: taoli@ifi.uzh.ch

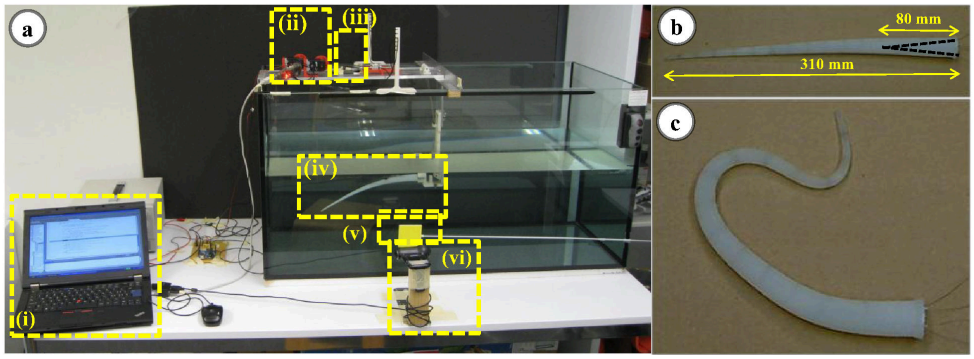


Fig. 1. (a) The experimental platform. It consists of a laptop PC (i), two servomotors (ii), with their rotation angles collected to the PC as sensory inputs. Also, one camera is used for motion recording (iii), and a soft robotic arm (iv), colored objects (white and yellow) to trigger behavior switching (v), and another camera proving visual input (vi) are prepared. (b) Structure of the soft robotic arm. The two dashed lines represent the fishing cables embedded in the arm. (c) The silicone robotic arm can be bent at any point along the arm in any direction.

structure proposed in [30], follows the morphology of an octopus arm. The total length of the cone-shaped soft arm (Fig. 1(b)) is 310 mm, with an active part of 80 mm from the base. The rest 230 mm is moved passively. As shown by the dashed lines in Fig. 1(c), the active part of the arm has two nonextensible fishing cables embedded inside. The two cables slant toward each other to meet at the arm's center. The soft arm is driven by alternately pulling the cables embedded in the active part of the arm, using two servomotors (Dynamixel^{TM} AX-12A+). The servomotors' angular positions are used as one type of sensory input, designated by integers from 0 to 1023. A camera (Logitech^{TM} Webcam Pro 9000) is placed in front of the water tank to provide visual input. Meanwhile, another camera of the same model is placed on the top of the platform to record the motion of the soft robotic arm.

A Java program implementing the learning and controlling strategy runs on a laptop PC to generate the sensorimotor loop. The program receives sensory input from position and visual sensors, described above, and generates the commands to the servomotors. According to the commands, the servomotors' angular positions are adjusted, designated by integer values from 0 to 1023, which correspond linearly to angles of 0 to 300 degrees. The units of position sensors are designated by [POS], which is about 0.29 degrees. An online learning method is implemented to enable the control program to develop the ability to embed and switch among different predefined behaviors according to sensory inputs. A timestep (T) is one sensing and actuation loop, or sampling period, of the control program. The settings of the network and the online learning procedure are detailed in the following sections.

The servomotor position, designated by integers between 0 and 1023, as described above, is adjusted according to commands to the motor regarding direction and speed. The servomotor speed (v) is the motor position change per timestep (T) and thus has the unit of [POS/T]. It can be set from 10 [POS/T] to 100 [POS/T] considering the limitation of the platform - a speed slower than 10 [POS/T] cannot exhibit the dynamics of the soft arm, while a speed faster than 100 [POS/T] would cause the servomotor to overheat or become damaged. In this paper, motor directions are set as binary values, $M = \{+1, -1\}$. If the direction gives $+1$ or -1, the motor is controlled to move from the current position toward the maximum position (L_{max}) or the relaxed position (L_{relax}) correspondingly with the speed (v). For each servomotor, L_{max} was determined so as not to cause the tip of the arm to touch the walls of the water tank during its movement. Note that the motor direction commands do not always take the motor's roller position to L_{max} or L_{relax}, but rather decides the motor's moving direction for each timestep. Also, if the direction command gives $+1$ or -1 when the current position is in L_{max} or L_{relax}, respectively, the position will stay unchanged for that timestep.

B. Task

When exploring control strategies inspired by the octopus, we have to embed various sequential behaviors. The aim of this study is to evaluate whether an online learning method for reservoir computing can be implemented on a soft robotic platform to embed multiple behaviors and achieve behavior switching by visual input. As explained in the INTRODUCTION section, an online learning method does not require a separate teaching data collection phase; thus, it has the potential to deal with unsuspected changes in the robot structure and the working environment. This potential is critical for physical robotic platforms implementation.

We attempted to embed three oscillatory behaviors for the soft robotic arm. The oscillatory behavior of the arm is achieved by alternately adjusting the tensions on the two cables embedded in the soft robotic arm. Initially, one cable (cable one) is in its relaxed state and the other (cable two) is in its maximum. Then, the motor driving cable one starts increasing the tension on the cable at a constant speed until

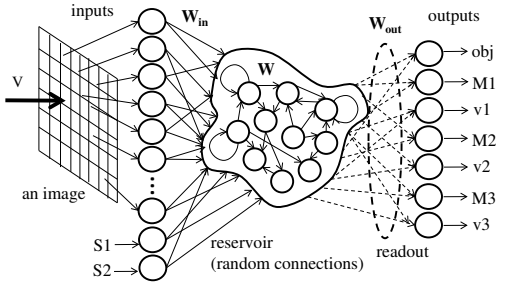


Fig. 2. Network architecture. There are 50 input nodes (position sensor inputs $S1$ and $S2$, visual input V of averaged greyscale values of each of the 48 equally divided rectangular areas of an image), 1000 reservoir neurons, and 7 output nodes (object obj and motor patterns $M1$, $v1$, $M2$, $v2$, $M3$, $v3$) to describe each of the three behaviors. All connections, except for the sparse internal connections of the reservoir, are fully connected but are not shown for simplicity. See text for details.

the cable reaches the predefined maximum position, while cable two is driven toward the relaxed position. Then, cable one, which is at its maximum position, starts to go back to the relaxed position, while cable two goes to its predefined maximum position. This alternate adjusting continues until a predefined timestep. We define three behaviors, which have the speeds of 30 [POS/t] (behavior one), 60 [POS/t] (behavior two), and 90 [POS/t] (behavior three), corresponding to visual inputs of a white object, no object, and a yellow object, respectively. For each of the three behaviors, the speed is the same for both cables and both directions.

After the online learning procedure, the ESN should be able to generate the motor commands to achieve the three oscillatory behaviors precisely according the visual input.

C. ESN and its online learning algorithm

1) Sensorimotor mapping: Fig. 2 shows the ESN used in this study. The sensorimotor loop describes the input and output of the ESN.

There are two types of sensory input. The first type is the servomotors' angular positions to represent the soft robotic arm dynamics. Because there is a position sensor for each of the two servomotors, we describe them as $S1$ and $S2$. Meanwhile, to switch among the three behaviors that are to be embedded in the ESN, we put different objects (Fig. 1(a)(v)) between the soft robotic arm (Fig. 1(a)(iv)) and the camera (Fig. 1(a)(vi)) as a trigger. To detect the presence of different objects, images of the object (if present), the soft robotic arm, and its surrounding environment are captured at the rate of one image per timestep. The image resolution is set to 640×480 pixels. We preprocess the image to use it as sensory input. First, the image is equally divided into 48 cells (6 in vertical and 8 in horizontal). The cell numbers are determined by considering the tradeoff between computational cost and the richness of information extracted from the image. Then the averaged grayscale values ($GrScl$) of each cell are taken as the second type of sensory input to the ESN. The grayscale value of a pixel is calculated from the corresponding R, G, and B values by $GrScl = 0.2989 * R + 0.5870 * G + 0.1140 * B$.

There are two types of reservoir output. The first type judges the object (obj) presented in front of the camera and its corresponding behavior. According to the object (obj), one of the three behaviors is selected. The mapping between the result of object recognition and the corresponding behavior is predefined. There is only one output node (obj) of the first type. After determining the object, the corresponding behavior needs to be mapped to actual motor commands. The second type of output is the motor patterns described by motor moving direction (M) and speed (v) for the three behaviors used in this study. For example, $M1$ and $v1$ describes the motor commands of moving direction and speed, respectively, for behavior one. There are six outputs of the second type for the three different behaviors. In summary, there are 50 reservoir inputs (two from position input and 48 from visual input) and seven outputs (one output describes the object and six outputs describe the three motor patterns) for the ESN.

2) Network architecture: The number of neurons in the reservoir is set to $N = 1000$ throughout the study. The network consists of three types of connection weights. The first type is the connection weights from the input nodes to the reservoir neurons (W_{in}, size 1000×50). The second type is the internal connection weights of the reservoir neurons (W, size 1000×1000). The last type is the connection weights from the reservoir neurons to the output nodes (W_{out}, size 7×1000). Connection weights W_{in} are randomly generated real values from [-0.005 0.005] and are fixed throughout the experiments. We used the method introduced in [25] to set W. The method can be summarized in three steps: (1) Randomly generate a sparsely connected matrix (W') with density $p = 0.1$; (2) Normalize W' to a matrix W'' with unit spectral radius by applying $W'' = W'/|\lambda_{max}|$, where $|\lambda_{max}| = \sqrt{p * N}$ is the spectral radius of W'; (3) Scale W'' to $W = \alpha W''$, where $\alpha < 1$; thus, W has a spectral radius of α. α is set to 0.85, which defines a network with echo state properties [25]. W_{out} is the weights that will be adapted in the training phase. We define the input sequence in n timesteps as $u(n) = (u_1(n), u_2(n), \cdots, u_{50}(n))$, where $u_1(n), u_2(n), \cdots, u_{48}(n)$ are the averaged gray scale values of the first, the second, $\cdots$, and the 48th cell of the 48 image cells; $u_{49}(n)$ and $u_{50}(n)$ are the sensory inputs of servomotor positions. To describe the states of the reservoir neurons, we use $x(n) = (x_1(n), x_2(n), \cdots, x_{1000}(n))$ and $r(n) = (r_1(n), r_2(n), \cdots, r_{1000}(n))$. $x(n)$ and $r(n)$ refer to the states of neurons before and after applying the hyperbolic tangent state transition function, respectively. The initial values of neuron states, $x(0) = (x_1(0), x_2(0), \cdots, x_{1000}(0))$, are random real numbers from [-0.5, 0.5]. The output of the network is $y(n) = (y_1(n), y_2(n), \cdots, y_7(n))$, where $y_1(n) = obj(n)$, $y_2(n) = M1(n)$, $y_3(n) = v1(n)$, $\cdots$, $y_6(n) = M3(n)$, and $y_7(n) = v3(n)$. Accordingly, the updating rules of the

reservoir states are defined as

$$g(x) = \tanh(x), \tag{1}$$
$$r^T(0) = g(x^T(0)), \tag{2}$$
$$x^T(n+1) = W_{in} * u^T(n+1) + W * r^T(n), \tag{3}$$
$$r^T(n+1) = g(x^T(n+1)), \tag{4}$$
$$y^T(n+1) = W_{out} * r^T(n+1). \tag{5}$$

3) Experimental procedure and online learning algorithm: To achieve the behavior switching, two phases are performed: learning and evaluating. In the learning phase, the motor output pattern is generated by the teaching signals. Meanwhile, the readout connection weights are trained online by a method called first-order reduced and controlled error (FORCE) learning [31], which is an error-based modification of the readout weights. An algorithm suitable for the FORCE learning should rapidly reduce and keep the error magnitude between the desired and actual output to a small value during the readout weights modification. The learning procedure terminates when it finds a set of fixed readout weights that can maintain a small error without further modification. In this study, we use the recursive least square (RLS) algorithm to implement the FORCE learning. For a given object (input), only the output connection weights to the related output node W'_{out} are updated. For instance, if the object is yellow, corresponding to behavior three, the connection weights to output nodes obj, $M3$, and $v3$ are updated during the learning. To illustrate, let $W_{out} = (W1_{out}, W2_{out}, \cdots, W7_{out})^T$, in which, $Wi_{out}(i = 1,2,\cdots,7)$ is a 1000×1 matrix and represents the output weights from the reservoir to output node i. Accordingly, $W'_{out} = (W1_{out}, W6_{out}, W7_{out})^T$ in the case of the yellow object. Note that the connection weights to output node obj are always updated during the learning phase because it reflects the presence of different objects. This procedure is described as follows:

$$e'(n) = y'(n) - f(n), \tag{6}$$
$$k(n) = P(n) * r^T(n+1), \tag{7}$$
$$c(n) = \frac{1.0}{1.0 + r(n+1) * k(n)}, \tag{8}$$
$$dW'_{out}(n+1) = e'^T(n) * k^T(n) * c(n), \tag{9}$$
$$W'_{out}(n+1) = W'_{out}(n) + dW'_{out}(n+1), \tag{10}$$
$$P(n+1) = P(n) - k(n) * (k^T(n) * c(n)) \tag{11}$$

in which, $P(n)$ is an $N \times N$ inverse correlation matrix with the initial value of $P(0) = I/\beta$ (β is a constant parameter and acts as the learning rate; we set $\beta = 1$); $f(n)$ is the desired (teaching) output with size 1×3; $y'(n)$ is the actual output; $e'(n)$ is the error. Both $y'(n)$ and $e'(n)$ are of size 1×3 and are defined similarly as W'_{out}.

When the error $e(n) = (e_1(n), e_2(n), \cdots, e_7(n))$ meets the condition that each of them is smaller than a threshold value (the thresholds of $e_1(n), e_3(n), e_5(n), e_7(n)$ were set as 0.01 and the thresholds of $e_2(n), e_4(n), e_6(n)$ were set as 0.25 in this study), the learning phase terminates and the evaluation phase begins. The error of each of the seven outputs is defined by the root-mean-square (RMS) error of the error values recorded in a sliding window to evaluate the learning result. At each timestep, the oldest error value is discarded and the latest one is added. To guarantee successful learning, the motor patterns should be completely presented. We set the error sliding window size at least two times the size of the motor pattern needed to be learned. For output Obj, the error sliding window size is set to 250 timesteps, which is the longest because it needs to evaluate the error during the presence of at least three objects. The window sizes for another six outputs are set to 50 timesteps, as the longest time series of the motor patterns is 22 (output $M1$). In the evaluation phase, the motor output pattern is generated by the ESN. We randomly present the visual input as one of the three cases (no object, yellow object, or white object) and observe the robotic platform output. The generated reservoir outputs and the desired outputs at each timestep are recorded for further analysis.

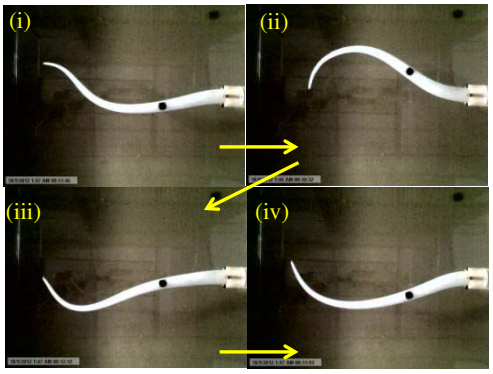


Fig. 3. A typical example of the soft arm motion. It shows a characteristic oscillatory behavior. The amplitude and frequency of the oscillation are different among behavior one, two, and three.

III. RESULTS

In this section, the behavior of the soft robotic arm is observed first. Then, the performance of the ESN to achieve embedding and switching among three behaviors by the visual input is evaluated. We also analyze the learning process by plotting the output errors during the learning and evaluation phases, as well as the trajectories of output weights amplitudes during learning.

A. Observations

One characteristic of the soft arm is the time delay in transmitting the movement of the arm generated by actuators from the arm base to the tip because the arm is soft. This is an intrinsic property of a soft body that is not observed in a rigid one, for example, in the case of a metal stick. As explained in the previous section, oscillatory behaviors are adopted and the speed of the arm oscillation is predefined from small to large in the order of behavior one, two, and

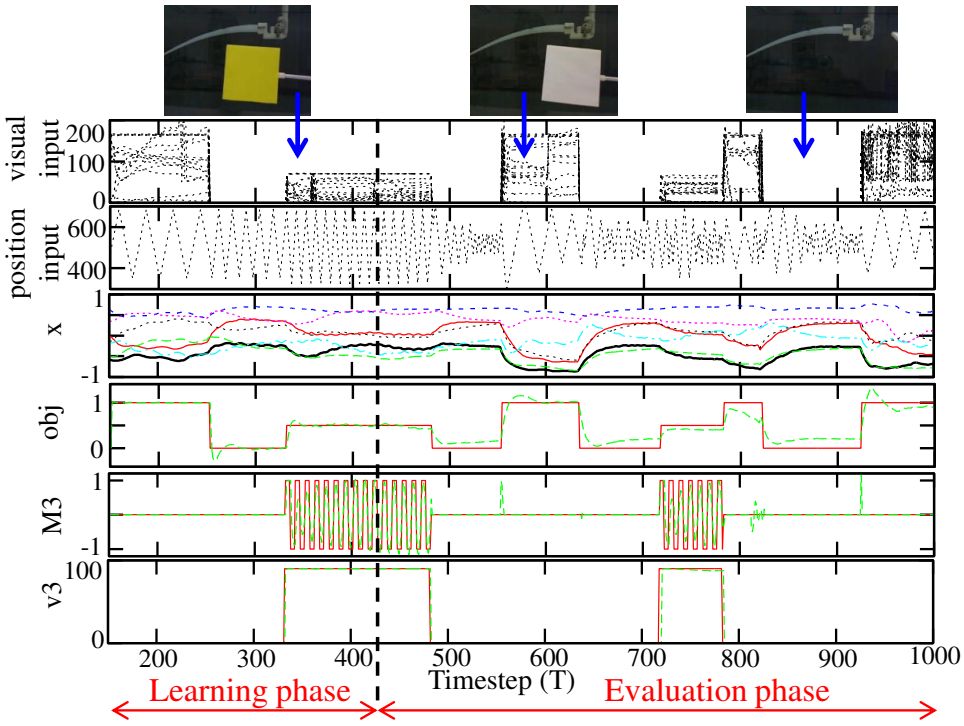


Fig. 4. An example showing the trajectories of the ESN variables. The trajectory contains both the online learning and the evaluation phases. The phase switches at timestep 412. From the upper to the lower line, three representative sample images taken from the camera are shown (objects are in rectangular shapes), as well as the trajectories of the sample visual inputs, position sensor inputs, several sample reservoir neuron activations (x), the ESN outputs to determine the behavior (obj), and the motor patterns ($M3$, $v3$) for behavior three (yellow object). In the plots showing the trajectories of the ESN output, the red solid lines show the target trajectories, whereas the green dashed lines show the ESN output trajectories. We can see that the output occasionally shows different values from the target but can be corrected using filters.

three. The diverse body dynamics of the soft arm, which is controlled by the ESN approach, is shown in Fig. 3.

B. ESN performance

The ESN performance is assessed by checking the output timeseries, the output RMS errors, and the magnitudes of readout weights vectors.

Fig. 4 shows an example of the trajectories of the ESN variables. Example visual input images, visual input time-series, position sensor inputs, several sample reservoir neuron activations, and three example outputs of the ESN (obj, $M3$, $v3$) during the online learning and evaluation phases are shown from top to bottom. The online learning phase terminates and the evaluation phase starts at timestep 412. The objects in front of the camera were randomly presented during the experiment. The timeseries show that the actual outputs agree well with the desired outputs, although some errors can be occasionally observed. These errors were corrected by a filter in the actual experiments.

The trajectory of the RMS errors of the seven outputs (obj, $M1$, $v1$, $M2$, $v2$, $M3$, $v3$) during the online learning

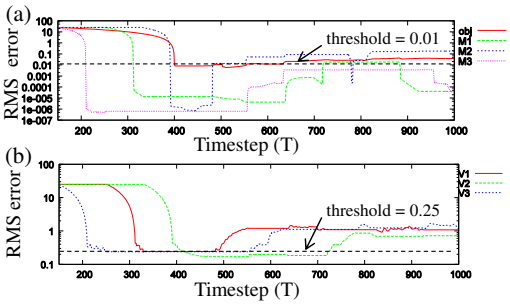


Fig. 5. Output errors of obj, $M1$, $M2$, and $M3$ (a), also of $v1$, $v2$, and $v3$ (b) during the online learning and evaluation phases corresponding to Fig. 4. The errors reduce dramatically in the online learning process and remain moderately small during the evaluation phase. Note that for each plot, the vertical axis is in logarithmic scale.

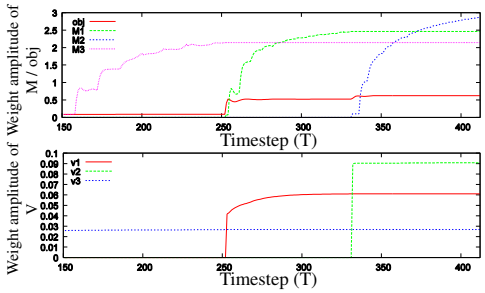


Fig. 6. The amplitudes of ESN output weights for output nodes, obj, $M1$, $M2$, and $M3$ (a), also $v1$, $v2$, and $v3$ (b) during the online learning phase of Fig. 4. All the weights get saturated during the learning phase and have small values. We observed that the change of the value for $v3$ were especially small.

and evaluation phases, corresponding to Fig. 4, are shown in Fig. 5. The plot shows that the RMS errors were dramatically reduced by the FORCE learning algorithm during online learning. Meanwhile, the RMS errors remained small, even though they increased a bit in the evaluation phase. The error plot shows that the online learning was successful.

Fig. 6 shows the output weight amplitudes for the seven outputs (obj, $M1$, $v1$, $M2$, $v2$, $M3$, $v3$) during the online learning phase, corresponding to Fig. 4. At each timestep, the output weight amplitudes of the seven output nodes are calculated by multiplying the output weight matrix by its transpose ($|W_{out}| = W_{out} * W_{out}^T$). Large output weight amplitudes after learning indicate the ESN involves cancelation between large positive and negative contributions. Such solutions tend to be unstable and sensitive to noise. It can be observed that the output weights amplitudes are not large and get saturated during the online learning phase. Small weight amplitudes mean that the ESN output comes from the proper combination of reservoir dynamics, and the performance should be stable.

IV. DISCUSSION AND CONCLUSION

This study shows that it is possible to embed and switch among multiple behaviors using an online learning method in a soft robotic platform. However, dealing with these oscillatory behaviors are not our ultimate purpose. These behaviors are merely used as study cases to test the feasibility of the online learning methods. Many interesting behaviors can be observed in the octopus. For example, in reaching for an object, the octopus uses bending propagation along the arm, and the octopus forms a joint-like structure in the arm when fetching an object. In future study, we will incorporate these behaviors.

In this paper, two phases are used to achieve embedding and switching multiple behaviors, namely online learning and evaluation. One interesting extension is to allow the control program go back to the online learning phase once the errors grow beyond pre-defined thresholds during the evaluation phase. In this way, the controller could potentially deal with changes in robot structure or the environment. Moreover, it seems straightforward to realize the learning tasks by adjusting only the readout weights during the learning of the ESN. But it is true that the performance also depends on parameters other than the readout, such as the number of nodes and the spectral radius of the reservoir. Many of these parameters need to be tuned by trial and error. A systematic method to tune the reservoir parameters is needed. These aspects will be studied in our future work.

REFERENCES

[1] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Applied Bionics and Biomechanics*, vol. 5, no. 3, pp. 99–117, 2008.

[2] E. Steltz, A. Mozeika, N. Rodenberg, E. Brown, and H. M. Jaeger, "Jsel: Jamming skin enabled locomotion," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, 2009, pp. 5672–5677.

[3] E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. R. Zakin, H. Lipson, and H. M. Jaeger, "Universal robotic gripper based on the jamming of granular material," *Proceedings of the National Academy of Science*, vol. 107, pp. 18 809–18 814, 2010.

[4] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, 2011.

[5] R. Pfeifer, M. Lungarella, and F. Iida, "The challenges ahead for bio-inspired 'soft' robotics," *Communications of the ACM*, vol. 55, pp. 76–87, 2012.

[6] P. Meier, M. Lang, and S. Oberthuer, "Reiterated tension testing of silicone elastomer," *Plastics, Rubber and Composites*, vol. 34, pp. 372–377, 2005.

[7] X. Nie, B. Song, Y. Ge, W. Chen, and T. Weerasooriya, "Dynamic tensile testing of soft materials," *Experimental Machanics*, vol. 49, no. 4, pp. 451–458, 2009.

[8] P. J. MacKerrow, *Introduction to robotics*. Sydney: Addison-Wesley, 1995.

[9] T. Gutnick, A. R. Byrne, B. Hochner, and M. Kuba, "Octopus vulgaris uses visual information to determine the location of its arm," *Current biology*, vol. 21 (6), pp. 460–462, 2011.

[10] Y. Gutfreund, T. Flash, Y. Yarom, G. Fiorito, I. Segev, and B. Hochner, "Organization of octopus arm movements: A model system for studying the control of flexible arms," *Journal of Neuroscience*, vol. 16 (22), pp. 7292–7307, 1996.

[11] Y. Gutfreund, "Patterns of arm muscle activation involved in octopus reaching movements," *Journal of Neuroscience*, vol. 18 (15), pp. 5976–5987, 1998.

[12] G. Sumbre, G. Fiorito, T. Flash, and B. Hochner, "Neurobiology: Motor control of flexible octopus arms," *Nature*, vol. 433(7026), pp. 595–596, 2005.

[13] R. Pfeifer and C. Scheier, *Understanding Intelligence*. MIT Press, 1999.

[14] R. Pfeifer and J. Bongard, *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press, 2006.

[15] R. Pfeifer, M. Lungarella, and F. Iida, "Self-organization, embodiment, and biologically inspired robotics," *Science*, vol. 318, no. 5853, pp. 1088–1093, 2007.

[16] K. Nakajima, T. Li, N. Kuppuswamy, and R. Pfeifer, "How to harness the dynamics of soft body: Timing based control of a simulated octopus arm via recurrent neural networks," *Procedia Computer Science*, vol. 7, pp. 246–247, 2011.

[17] ——, "Biologically inspired control of a simulated octopus arm via recurrent neural networks," in *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation (GECCO '11)*, 2011, pp. 21–22.

[18] ——, "Harnessing the dynamics of a soft body with "timing": Octopus inspired control via recurrent neural networks," in *2011 15th International Conference on Advanced Robotics (ICAR)*, 2011, pp. 277–284.

[19] K. Nakajima, T. Li, and R. Pfeifer, "Timing and behavioral efficiency in controlling a soft body: A case study in octopus reaching behavior," in *2th International Conference on Morphological Computation (ICMC)*, 2011, pp. 132– 134.

[20] K. Nakajima, T. Li, H. Sumioka, M. Cianchetti, and R. Pfeifer, "Information theoretic analysis on a soft robotic arm inspired by the octopus," in *2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011, pp. 110–117.

[21] K. Nakajima, T. Li, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, "Local information transfer in soft robotic arm," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 1273–1280.

[22] T. Li, K. Nakajima, M. Kuba, T. Gutnick, B. Hochner, and R. Pfeifer, "From the octopus to soft robots control: an octopus inspired behavior control architecture for soft robots," *Vie et Miliew/ Life and Environment*, vol. 61 (4), pp. 211–217, 2012.

[23] J. Kuwabara, K. Nakajima, R. Kang, D. T. Branson, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, "Timing-based control via echo state network for soft robotic arm," in *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012, pp. 2943–2950.

[24] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, pp. 78–80, 2004.

[25] H. Jaeger, "Tutorial on training recurrent neural networks, covering bptt, rtrl, ekf and the "echo state network" approach," German National Research Center for Information Technology, Tech. Rep. 159, 2002.

[26] B. Schrauwen, D. Verstraeten, and J. V. Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th European Symposium on Artificial Neural Networks (ESANN2007)*, 2007, pp. 471–482.

[27] W. Maass, T. Natschlaeger, and H. Markram, "Real-time computing without stable states: a new framework for neural computation based on perturbations," *Neural Computation*, vol. 14(11), pp. 2531–2560, 2002.

[28] T. Li, K. Nakajima, M. Cianchetti, C. Laschi, and R. Pfeifer, "Behavior switching by using reservoir computing for a soft robotic arm," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 4918–4924.

[29] T. Li, K. Nakajima, M. Calisti, C. Laschi, and R. Pfeifer, "Octopus-inspired sensorimotor control of a multi-arm soft robot," in *2012 International Conference on Mechatronics and Automation (ICMA)*, 2012, pp. 948–955.

[30] M. Cianchetti, A. Arienti, M. Follador, B. Mazzolai, P. Dario, and C. Laschi, "Design concept and validation of a robotic arm inspired by the octopus," *Materials Science and Engineering C*, vol. 31, pp. 1230–1239, 2011.

[31] D. Sussillo and L. F. Abbott, "Generating coherent patterns of activity from chaotic neural networks," *Neuron*, vol. 63, pp. 544–557, 2009.

Appendix E

Octopus-Inspired Sensorimotor Control of a Multi-Arm Soft Robot

Li, T., Nakajima, K., Calisti, M., Laschi, C., and Pfeifer, R. (2012). Octopus-Inspired Sensorimotor Control of a Multi-Arm Soft Robot, in *2012 IEEE International Conference on Mechatronics and Automation*, pp. 948–955.

This is a version that was accepted for publication. Final version of the article can be found at ieeexplore.ieee.org (doi: 10.1109/ICMA.2012.6283271).

Octopus-Inspired Sensorimotor Control of a Multi-Arm Soft Robot

Tao Li*, Kohei Nakajima*, Marcello Calisti†, Cecilia Laschi†, and Rolf Pfeifer*
*Department of Informatics, University of Zurich, 8050 Zurich, Switzerland
Email: *taoli@ifi.uzh.ch*
†The BioRobotics Institute, Scuola Superiore Sant'Anna, 56026 - Pontedera (PI), Italy

***Abstract*—Soft robots have significant advantages over traditional rigid robots because of their morphological flexibility. However, the use of conventional engineering approaches to control soft robots is difficult, especially to achieve autonomous behaviors. With its completely soft body, the octopus has a rich behavioral repertoire, so it is frequently used as a model in building and controlling soft robots. However, the sensorimotor control strategies in some interesting behaviors of the octopus, such as octopus crawling, remain largely unknown. In this study, we review related biological studies on octopus crawling behavior and propose its sensorimotor control strategy. The proposed strategy is implemented with an echo state network on an octopus-inspired, multi-arm crawling robot. We also demonstrate the control strategy in the robot for autonomous direction and speed control. Finally, the implications of this study are discussed.**

***Index Terms*—octopus; crawling behavior; sensorimotor control; multi-arm soft robot; reservoir computing.**

I. Introduction

Soft robots research is one of the frontiers and trends in biomimetic robotics [1]. Because of the morphological flexibility and the interaction safety of soft robots, they have attracted considerable interest for their practical applications, such as rescue robots that could crawl through rubble [2] or as flexible manipulators [3]. However, one challenge in soft robotics research is to achieve autonomous behaviors. The behaviors of animals and robots are well recognized to be shaped by the interaction among the brain (control), the body (material and structure), and the environment [4][5]. In the case of soft robots, however, designing the robot body and the control strategy remains a significant challenge. Roboticists therefore turn to nature for inspiration.

The octopus has surprising abilities to control its totally soft body for locomotion [6], reaching [7], and even precise point-to-point fetching movements [8], to name a few. Its totally soft body enables it to squeeze through gaps much smaller than its body. Therefore, it is an ideal model for roboticists [9][10][11]. From our review of biological studies, the control mechanisms of some interesting octopus behaviors are still unclear, one such case is the octopus crawling behavior. In this paper, we use a multi-arm soft robot to study a proposed sensorimotor control strategy of octopus crawling behavior. The main contributions of this work are as follows: (1) We propose a sensorimotor control strategy for octopus crawling behavior by synthesizing available biological studies on octopus behavior, sensory systems, and neurophysiology; (2) An octopus-inspired multi-arm robot is built as an experimental platform to examine and demonstrate octopus crawling behavior; (3) Finally, the proposed control strategy is implemented and demonstrated on the octopus-inspired, multi-arm robotic platform with the use of an echo state network (ESN) [12], which has been shown to control soft robots effectively [13][14].

The rest of this paper is organized into four sections. First, we review related biological investigations and propose a sensorimotor control strategy for octopus crawling behavior. Second, the robotic experimental platform used to study the octopus crawling behavior is described. Third, the specification of the robotic platform and the performance of the control strategy are analyzed. Finally, we conclude the study and discuss the implications.

II. Sensorimotor control strategy of the octopus crawling behavior

The anatomy and the functionality of the octopus nervous systems are well documented, but the sensorimotor control of octopus crawling behavior remains largely unknown. Octopus crawling behavior involves processing of sensory information and coordination of multiple arms by the nervous systems. In this section, we briefly review octopus crawling behavior and the role of the sensory and nervous systems in this behavior. Then we propose a sensorimotor control mechanism of octopus crawling behavior.

A. Crawling behavior

Crawling is one of the commonly observed but rarely studied octopus locomotion modes [6][15]. A recent study showed the basic pattern of octopus crawling movement [10]. Typically, the octopus uses its arms both toward and opposite its intended moving direction when crawling: it chooses arms opposite to the moving direction as crawling arms, and arms toward the moving direction as probes. The crawling arms

*This work is supported by the European Commission in the ICT-FET OCTOPUS Integrating Project (EU project FP7-231608).

shrink before touching the ground, firmly hold the ground with the use of suckers at a region about one third from the tip of the arm, and finally extend the section between the catching location and the arm base to provide pushing forces. The crawling arms move alternatively and push the body forward. In case the octopus intends to change its crawling direction, it selects the appropriate arms not in use. Meanwhile, the octopus has no arm preference but uses all its arms equally in the crawling movement because of their symmetrical arrangement.

B. Sensory systems

The octopus has rich sensory systems. Its visual system is critical for the sensorimotor control of its crawling behavior. The animal employs a sophisticated, bilaterally symmetrical visual system, which consists of two eyes placed laterally on the head [16]. Its potential field of vision is 360 degrees, with a binocular vision of 5 degrees anterior and posterior of its head [17]. Therefore, the animal can visually guide its crawling toward an object located in any direction. Byrne et al. ([18][19]) found that octopus arm choice is strongly influenced by eye use and the animal mostly uses the arm that is in a direct line between the object and the eye used. As the direction of approaching is visually guided, the octopus arms used for crawling are spatially determined by visual information. Moreover, the extensive visual system of the octopus consists of more than 20 percents of the total neurons of the octopus nervous system. Thus, the visual system could preprocess visual input and reduce the workload of the octopus brain. The flexible octopus arms also have tactile and proprioceptive sensing abilities [17][20][21] [22]. The octopus mainly uses these two sensory modalities to locally control the movements of individual arms.

C. Nervous systems

A distribution of functionalities exists between the octopus central nervous system (CNS) and the peripheral nervous system (PNS) [23]. Aside from integrating preprocessed visual input from the visual system, the relatively small CNS controls the highly autonomous PNS. The CNS has been shown to only send movement initiation signals to the PNS [24]. The PNS takes a large part of motor control function by embedded local motor programs [25].

D. Sensorimotor control strategy

Fig. 1 summarizes the flow of information in octopus crawling behavior on the basis of the biological studies reviewed in the previous subsections. Accordingly, sensorimotor control strategy of the octopus crawling behavior could be summarized as follows: (1) The octopus visual system (Fig. 1(i)) takes and preprocesses visual input, which covers 360 degrees of the environment, and sends the visual information to the CNS (Fig. 1(ii)). (2) The octopus CNS determines crawling direction and which arm(s) to use, and then it sends initiation signals to the PNS (Fig. 1(iii)). (3) According to the signals sent by the CNS, the PNS controls individual arms by using locally embedded motor programs to achieve the crawling behavior.

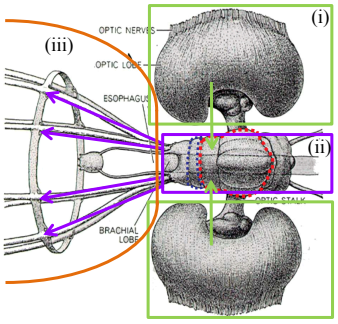


Fig. 1. Information flow in the octopus crawling behavior. The octopus visual system (i) takes and preprocesses visual input and sends it to the CNS (ii); the octopus CNS (ii) sends initiation signals to the octopus PNS (iii), according to the input; the octopus PNS (iii) controls the octopus arms for crawling. (Note: the background figure of the octopus nervous system is adapted from [17].)

In the next section, we present the experimental setting to implement and evaluate this sensorimotor control strategy on a robotic experimental platform.

III. Experimental Setting

The experiment aims to evaluate if the proposed control strategy of octopus crawling behavior could be implemented in a multi-arm soft robotic platform. The experimental platform setup, sensorimotor loop and visual input, controller architecture, task, and experimental procedure are presented below.

A. Platform setup

The platform setup is shown in Fig. 2(A). The platform consists of a six-arm soft robot (Fig. 2(B)), its motor control and data acquisition electronics, a control program running on a PC, a colored balloon, and a swimming pool containing fresh water as the working environment.

The crawling robot, crawler, used in this study is the extension of a multi-arm soft robot introduced in [10][27][28]. On top of the crawler is a mini omnidirectional camera that mimics the 360-degree visual system of the octopus; this camera has a weight of 110 grams and a height of 12 cm. The lower part of the camera is covered by two layers of balloons to isolate the water environment. The image module is a VPC-798 CCD board camera from the Pacific Corporation. The board camera connects to a PC USB port through an EasyCAP USB 2.0 image capture adaptor. Under the camera is a block of plastic foam, used to install the camera and to provide the buoyancy force. Six identical cone-shaped silicone arms

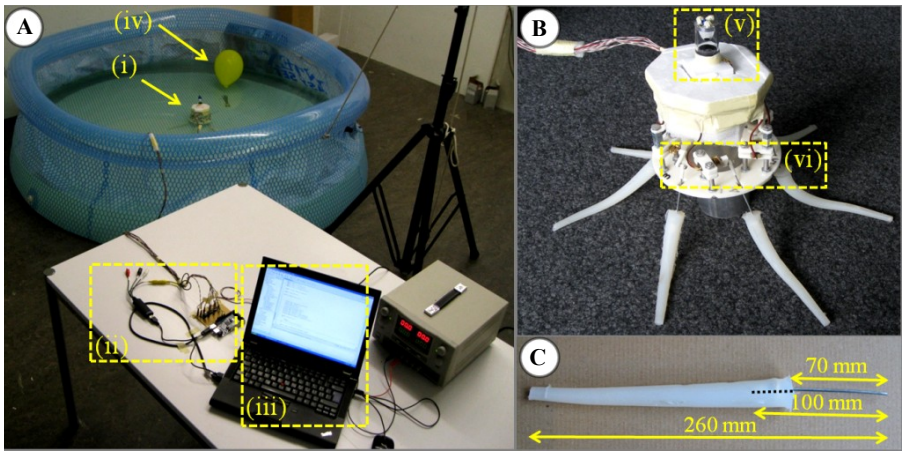


Fig. 2. Experimental platform. (A) Platform setup. The platform consists of a six-arm soft robot, crawler (i), electronics (ii), a laptop PC (iii), a colored balloon (iv), and a swimming pool containing fresh water as the working environment. (B) The crawler. It mainly contains an omnidirectional camera (v) and six equally distributed silicone arms. Each arm is driven by a DC motor through a crank and connecting rod mechanism (vi). (C) A silicone arm. Each of the corn-shaped silicone arms contains a flexible steel cable embedded in its proximal part.

are attached evenly around the base plate. Each arm (Fig. 2(C)) is made of commercial silicone (ECOFLEX^{TM} 00-30) with a steel cable embedded in its proximal part. Each of the silicone arms is driven by a DC motor (GM 12a with a 120:1 gearbox) through a crank and connecting rod mechanism. The DC motors are controlled by an Arduino^{TM} Mega 2560 board through Pulse Width Modulation (PWM) outputs and powered by a DC motor driver board. The motors can be controlled to stop, or to rotate in two different speeds and in both directions, clockwise (CW) and counterclockwise (CCW).

With proper designing of the dimensions of the crank and connecting rod mechanism, each actuated silicone arm provides pulling or pushing movements for the crawler depending on the motor rotation direction [27]. Interestingly, the robot can crawl in the swimming pool by simply rotating the DC motors. This ability is attributed to the outsourcing of a part of the control functionality to silicone material properties (e.g., the motion seems to be generated by the friction between the silicone arm and the ground) and to the properly designed crank and connecting rod mechanism [4][5]. The previous section showed that the PNS controls octopus arm motions, such as arm shrinking or extending. In our platform, the role of the PNS is embedded in the body property. By controlling the rotation speed of the motor, the robot can change the speed of crawling behavior. To move to a certain direction (D), the octopus uses the arm opposite to such direction. Meanwhile, the arm towards the crawling direction is also set to move to assist the crawling motion [1]. The former arm provides pushing by moving its DC motor CCW, whereas the latter arm provides pulling by driving its corresponding DC motor CW. The water level of the swimming pool and the supply voltage of the DC motors are also experimentally adjusted (see the RESULTS section) to make the crawler move efficiently in the pool. With the omnidirectional camera outside of the water, the crawler floats in the water, and only its arms touch the bottom of the swimming pool during the experiment.

The control program (in *Java*) running on a laptop PC embeds the function of the octopus CNS. It receives and processes the visual input from the omnidirectional camera and sends out the control parameters (moving direction and speed) to the Arduino^{TM} board through a series port. The unit of timestep in this paper is a sensing and actuation loop of the control program. An ESN is implemented (and trained) to generate the sensory-motor loop. The *RGB* values of the captured images are used as sensory inputs (see the next subsection for the details), whereas the moving direction ($D1, D2, \cdots, D6$) and the velocity (v) of the motor rotation are used as outputs. All outputs take binary values, $+1$ or -1. For the moving direction, $+1$ means the direction is selected, whereas -1 means it is inhibited. Only one direction should

[1]Note that the use of arms in biological octopus crawling behavior is different, as described in our review of biological studies. In the present work, we used a different arm combination to avoid occasional overturns of the crawler. However, mapping of the desired crawling behavior (direction and speed) and the motor commands was predefined and implemented in the Arduino^{TM} board. Thus, this is not a limitation of the proposed control strategy.

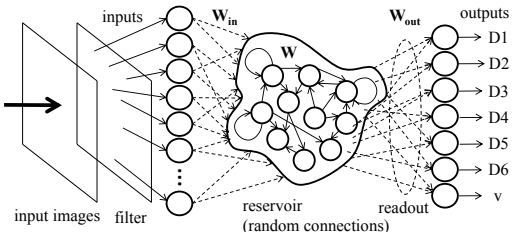


Fig. 3. Network architecture used in this paper. There are 2304 input nodes (sensory inputs; averaged R, G, and B values of each of the 768 equally divided rectangular areas of the images), 300 reservoir neurons, and 7 output nodes (crawler moving directions ($D1$, $D2$, $\cdots$, $D6$), and velocity (v)). All connections are fully connected but are not shown for simplicity. See text for details.

take a $+1$ value, whereas the others should take a -1 value. For the speed, $+1$ means high speed, whereas -1 means low speed. For instance, an output of $(D1 = -1, D2 = +1, D3 = -1, D4 = -1, D5 = -1, D6 = -1, v = +1)$ means moving to direction 2 in high speed. The detailed setting and training of the ESN are explained in the following subsections.

B. Sensorimotor loop and visual input

Sensorimotor loop describes the input and the output of the ESN. As explained previously, the *RGB* information of the processed images taken from the omnidirectional camera is used as visual input. Images of the surrounding environment are captured at the rate of one image per second, which is sufficient because of the limited moving speed of the crawler. The image resolution is set to 640×480 pixels. We apply a fixed *RGB* color filter ($R = 159$, $G = 151$, $B = 86$, $tolerance = 40$) to remove most of the background scenes from the image, and we leave the balloon. Then the filtered image is equally divided into 768 cells (24 in vertical, 32 in horizontal). We determine the cell numbers by considering the tradeoff between computational cost and the richness of information extracted from the images. We take the average R, G, and B values of each cell as the sensory input to the ESN. The crawler's six possible moving directions (D) and two possible moving velocities (v) are used as the ESN output. The moving direction and the speed are finally mapped to the PWM commands, as described in the previous section. In summary, there are 2304 reservoir inputs and 7 outputs for the ESN (Fig. 3).

C. ESN architecture

Fig. 3 shows the ESN used in this study. The number of neurons in the reservoir is set to 300 throughout the study. The network consists of three types of connection weights. The first type is the connection weights from the input nodes to the reservoir neurons (W_{in}, size 300×2304). The second type is the internal connection weights of the reservoir neurons (W, size 300×300). The last type is the connection weights from the reservoir neurons to the output nodes (W_{out}, size 7×300). Connection weights W_{in} are randomly generated real values from [0.0 0.002] and are fixed throughout the experiments. We used the method introduced in [12] to set W. The method can be summarized as what follows: (1) Randomly generate an internal weight matrix (W'). (2) Normalize W' to a matrix W'' with unit spectral radius by applying $W'' = W'/|\lambda_{max}|$, where $|\lambda_{max}|$ is the spectral radius of W'. (3) Scale W'' to $W = \alpha W''$, where $\alpha < 1$; thus, W has a spectral radius of α. α is set to 0.9, which is determined heuristically. W_{out} is the weights that will be adapted in the training phase. We define the input sequence in n timesteps as $u(n) = (u_1(n), u_2(n), \cdots, u_{2304}(n))$, where $u_1(n)$, $u_2(n)$, and $u_3(n)$ are the averaged R, G, and B values of the first of the 768 cells. The state of the reservoir neurons is $x(n) = (x_1(n), x_2(n), \cdots, x_{300}(n))$. The output of the network is $y(n) = (y_1(n), y_2(n), \cdots, y_7(n))$, where $y_1(n) = D1(n)$, $y_2(n) = D2(n)$, $\cdots$, $y_6(n) = D6(n)$, and $y_7(n) = v(n)$. The updating rules of the connection weights are defined as

$$x^T(n+1) = g(W_{in} * u^T(n+1) + W * x^T(n)), \quad (1)$$

$$y^T(n+1) = W_{out} * x^T(n+1), \quad (2)$$

$$g(x) = \tanh(x). \quad (3)$$

Note that we actually applied $g(x)$ to calculate $y(n+1)$ because they are binary values. Next, we need to decide a teacher output $(D(n) = (D_1(n), D_2(n), \cdots, D_6(n)), v(n))$ in order to determine W_{out}. As we will explain in the following section, $D(n)$ is determined according to the position of the target (balloon). The reservoir states, $x(n)$ for $n = t_{start}, t_{start}+1, \cdots, t_{start}+t_{train}$ are collected into the rows of a state-collecting matrix X of size $t_{train} \times 300$, where t_{start} is the timestep to start collecting the data, and t_{train} is the timesteps used for the training data. The teacher outputs $D(n)$ are collected into the rows of a matrix T of size $t_{train} \times 7$. Therefore, we directly obtain the desired weights by multiplying the pseudoinverse of X (X^*) with T:

$$W_{out} = X^* T. \quad (4)$$

D. Task

This study aims to achieve autonomous direction and speed control of the robotic crawler by implementing the proposed sensorimotor control strategy of octopus crawling behavior. The robot should move toward the colored balloon (target) by using its arms in line with the crawling direction and by adjusting its crawling speed according to the target distance, i.e., if the robot is outside of the threshold distance ($d_{threshold}$), it should approach the target with high speed (v_{high}); otherwise, it should approach the target with low speed (v_{low}), in which v_{high} and v_{low} are fixed and defined heuristically to clearly show the change in speed. The task setting is illustrated in Fig. 4.

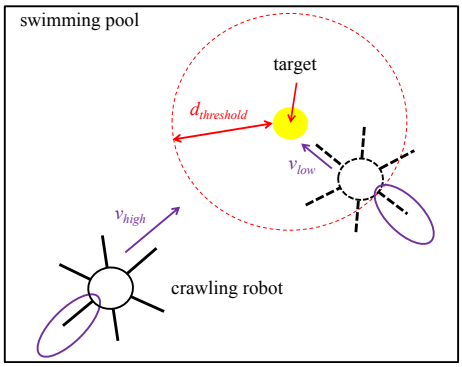


Fig. 4. Schematic diagram explaining the task in this study. The crawler should approach the target by using the proper arms and by changing its speed according to the distance to the target (use high speed (v_{high}) if the crawler is outside of the threshold distance ($d_{threshold}$); otherwise, a low speed (v_{low}) should be used.

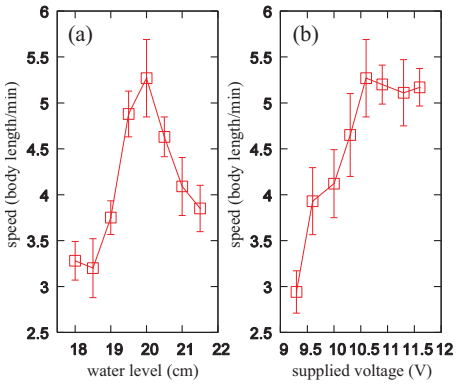


Fig. 5. Crawler performance quantification. (a) shows the influence of crawler speed according to the water level. The crawler speed has a maximum (optimal) value of 5.27 [body length/min] when the water level is 20 cm because of a good balance between the load and the propulsion force. (b) shows the relationship between the crawler speed and the supplied voltage. The crawler speed saturates when the supplied voltage is higher than 10.6 V because of the limited contact force and friction between the crawler arms and the bottom of the swimming pool. For (a) and (b), five trials are tested in each condition for 20 timesteps. The error bar represents the standard deviation.

E. Experimental procedure and network training

Three phases are used for the experiment: teaching, learning, and evaluating. In the teaching phase, we adopted a synthetic approach. By actually navigating the crawler to the target with manually generated motor outputs, we collected a set of teaching data (visual images as inputs and corresponding motor outputs) for 3000 timesteps (250 timesteps for each of the 6 moving directions and the 2 speeds). At each timestep, an image of the surrounding environment is captured by the omnidirectional camera and saved. Then we implement the learning phase by using the data generated in the teaching phase to train the linear reservoir readout connection weights. After training, the ESN is evaluated to switch among the six directions and the two speeds according to the target position. During the evaluation, we first put the target at a random position at a distance beyond the $d_{threashold}$ in the swimming pool. The crawler should approach the target by generating proper moving direction and speed. The generated moving direction $(D_1(n), D_2(n), \cdots, D_6(n))$ and the velocity $v(n)$ at each timestep are recorded. Meanwhile, the captured images at each timestep are saved to determine the desired moving direction and velocity for evaluation.

IV. RESULTS

In this section, we quantify the specification of the crawler in terms of moving speed. The behavior of the crawler equipped with the proposed control strategy is observed. Then, we evaluate the reservoir performance in detail by checking the timeseries. Finally, the performance of the controller is analyzed.

A. Crawler performance quantification

Before the actual experiment, we evaluated two major factors that affect the crawler's speed: water level and power supply voltage. As described in the platform setup, the crawler floats in the water, with the arms touching the bottom of the pool. The direct driving force of the crawler is the friction between the ground and the silicone arms. Therefore, checking how the water level and the supplied motor voltage affect the friction force and thus the motor speed is important. The relation between the speed and the water level is shown in Fig. 5(a). We can see that (1) the speed initially increases with the water level, and (2) the speed starts to descrease after a peak at the water level of 20 cm. The initial speed increase is due to the increasing water level that moves the crawler upward and thus reduces the load on the arms. The subsequent speed decrease is due to the difficulty of the arms to touch the bottom of the swimming pool, and the propulsion force reduces correspondingly when the water level is more than 20 cm. This relationship shows the tradeoff between the load and the propulsion force of the crawler determined by the water level through the buoyancy force. Fig. 5(b) shows the relation between the crawler speed and the supplied voltage of the DC motors which drive the arms. We can see that the speed initially increases with the supplied voltage and then becomes saturated when the voltage is higher than 10.6 V. The initial increase is due to the increase in the speeds of the DC motors with the supplied voltage. However, the slip on the bottom of the swimming pool because of the limited

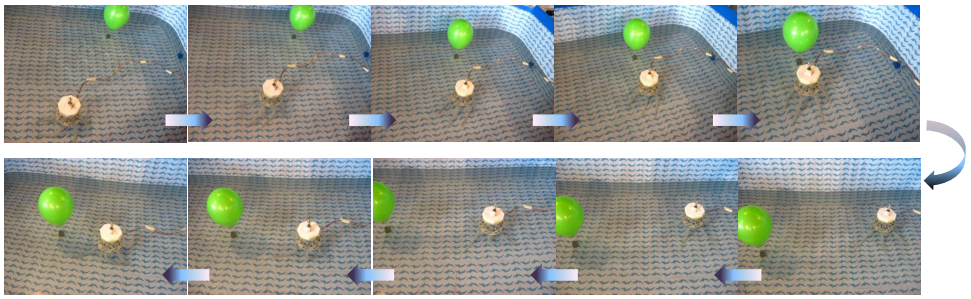


Fig. 6. Typical examples of the crawler behavior. The upper line and the lower line show the crawler approach two different target locations. Time evolves from the left to the right in the upper line and continues in the lower line from the right to the left. We can see that the moving direction of the crawler changes according to the target (balloon) locations. Images are extracted from a video recording at a 10-second interval.

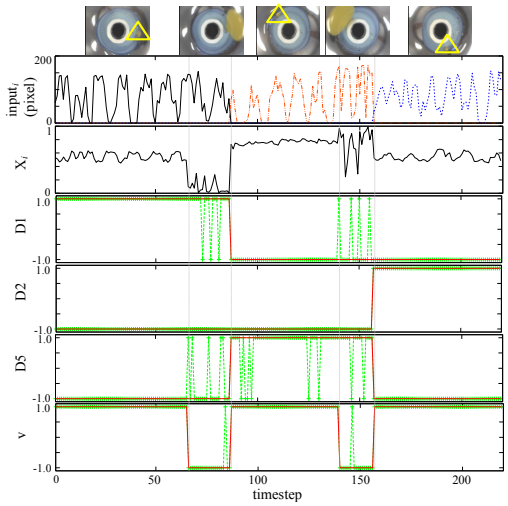


Fig. 7. Examples showing the trajectories of the ESN variables. From the upper to the lower line, five representative sample images taken from the omnidirectional camera are shown (yellow triangles indicate the targets), as well as the trajectories of the sample sensor inputs, one sample reservoir neuron activation (x), the ESN outputs of the directions ($D1$, $D2$, $D5$), and the velocity (v). In the plots showing the trajectory of the direction and the velocity output, the red line shows the target trajectory, whereas the green line shows the ESN output trajectory. We can see that the output only occasionally shows different values from the target.

contacting force and the increased water drag force to the crawler limit the speed increase. From these two tests, we used a water level of 20 cm and a power supplier voltage of 10.6 V in the experiments.

B. Observations

The autonomous switching of the moving direction of the crawler according to the target position is shown in Fig. 6. We can see from this figure that the crawler successfully follows the changing position of the target, although the speed change cannot be observed in the image sequence. We also observed in the experiment that the crawler mostly approaches the target straightly with the use of the same pair of arms in line with the crawling direction. However, the crawler occasionally needs to change the arms it uses because of the deviation from the desired crawling direction. This deviation is due to the stop of the DC motors at random positions and to the occasional interference to the arms in crawler movement.

C. ESN performance

Fig. 7 shows an example of the timeseries of visual input, neural activation, and four related outputs of the ESN when the crawler is autonomously running in the pool. The position of the target is changed three times during the run. The timeseries show that the actual outputs agree well with the desired outputs, although some errors can be occasionally observed.

To evaluate the generalization capacity of the network, we selected a typical ESN after training, and by simulating the visual input, we checked the performance of both “direction outputs" and “velocity output". We simulated the network inputs corresponding to each of the 768 cells with the value (R,G,B) = (159, 151, 86) and checked their response for each case to evaluate the performance of the direction outputs. When we stimulate each cell, the neighboring two cells (“radius 2") are also stimulated with the same input value. As the reservoir is a dynamical system, we used the response of the direction outputs after 10 timesteps for the evaluation. Fig. 8(a) shows the performance of the direction output $(D1, D2, \cdots, D6)$ according to the simulated object positions. We can see that the network could reliably output

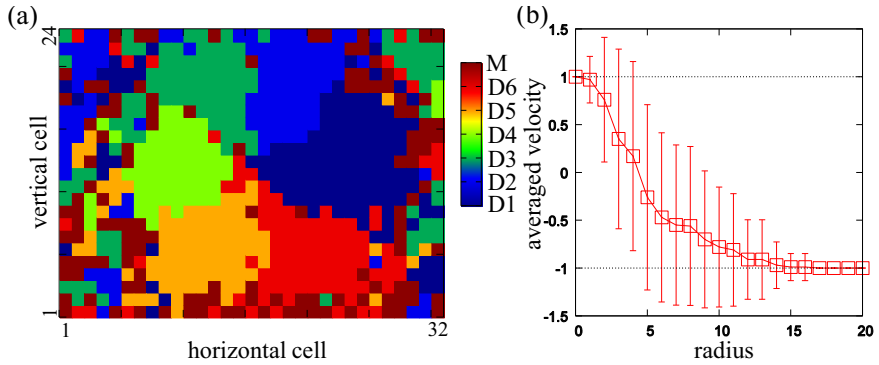


Fig. 8. Performance of the network. (a) shows the performance of the direction outputs according to the stimulated cell position. We can clearly see that the network successfully outputs the corresponding direction to the stimulated cell position. If more than two direction outputs show "+1" or all the direction outputs show "−1", then we call the response a mistake (M), which is represented in brown color in the plot. (b) shows the averaged velocity output over 200 randomly stimulated visual cells by varying the size of the radius for the stimulations. The error bar represents the standard deviation. We can clearly see the transition of the averaged value from "+1" to "−1" according to the increase in radius size. This result means that the velocity output is sensitive to the size of the stimulated region, which corresponds to the distance to the target.

the correct moving direction, whereas the fluctuated responses are observed in a region corresponding to the scene outside the pool. This response is reasonable, as the target will not appear outside the pool during the experiment; therefore, the controller was not trained for this region.

Next, we evaluated the performance of the velocity output. As the velocity output should be sensitive to the target distance and hence the target size on the image, we varied the size of the stimulated region of the input cells from radius 0 to 20 (cells) and observed how the performance of the velocity output depends on the radius size. For each radius size, we selected 200 random cells to stimulate in the image and averaged 200 corresponding velocity outputs. Fig. 8(b) shows the relationship between the averaged velocity output and the radius size. We can see that the velocity output gradually transits from high (+1) to low (−1) as the radius size becomes larger. This result means that the velocity output responds well to the visual input.

V. CONCLUSIONS AND DISCUSSIONS

In this study, we proposed a sensorimotor control strategy of the octopus crawling behavior, implemented it on an experimental robot platform, and achieved autonomous direction and speed switching.

As shown in the RESULTS section, the robot commonly uses the same pair of arms in line with the intended moving direction, but it occasionally needs to change the arms used to correct the moving direction. As the octopus has a potential vision field of 360 degrees, it does not have to change its heading to move to different directions. A significant tendency for heading change, if observed, may suggest that the animal has a preference for arm use. Meanwhile, an occasional heading change would be the result of unavoidable deviation in its moving direction because of disturbance from the environment, as observed in the robot experiment.

An area of further research for us can be the identification of the potential advantages of the ESN through the design of a more challenging task. As the task used in the present work is not quite difficult, the use of an alternative method such as a look-up table can be argued to achieve the same result. However, we believe that for more difficult tasks that need memory or context dependent information, the same ESN would still achieve satisfactory performance, whereas the look-up table will fail. We will test the controller for advanced tasks in our future study.

ACKNOWLEDGMENT

T. L. and K. N. thank Dr. Davide Scaramuzza for helpful discussions on the *RGB* filter design.

REFERENCES

[1] D. Trivedi, C. D. Rahn, W. M. Kier, and I. D. Walker, "Soft robotics: Biological inspiration, state of the art, and future research," *Appl. Bionics Biomech.*, vol. 5, pp. 99 - 117, 2008.

[2] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *P. Natl. Acad. Sci.*, 2011.

[3] E. Brown, N. Rodenberg, J. Amend, A. Mozeika, E. Steltz, M. R. Zakin, H. Lipson, and H. M. Jaeger, "Universal Robotic Gripper Based on the Jamming of Granular Material," *P. Natl. Acad. Sci.*, vol. 107, pp. 18809-18814, 2010.

[4] R. Pfeifer, M. Lungarella, and F. Iida, "Self-Organization, Embodiment, and Biologically Inspired Robotics," *Science*, vol. 318, pp. 1088-1093, 2007.

[5] R. Pfeifer and J. Bongard, *How the body shapes the way we think: a new view of intelligence*. Cambridge, Mass.: MIT Press, 2006.

[6] C. L. Huffard, F. Boneka, and R. J. Full, "Underwater Bipedal Locomotion by Octopuses in Disguise," *Science*, vol. 307, pp. 1927-, 2005.

[7] Y. Gutfreund, "Patterns of arm muscle activation involved in octopus reaching movements," *J. Neurosci.*, vol. 18, no. 15, pp. 5976-5987, 1998.
[8] G. Sumbre, G. Fiorito, T. Flash et al., "Octopuses Use a Human-like Strategy to Control Precise Point-to-Point Arm Movements," *Curr. Biol.*, vol. 16, no. 8, pp. 767-772, 2006.
[9] E. Guglielmino, N. Tsagarakis, and D. G. Caldwell, "An octopus anatomy-inspired robotic arm," in *Proceedings of the 2010 International Conference on Intelligent Robots and Systems (IROS'10)*, pp. 3091-3096, 2010.
[10] M. Calisti, M. Giorelli, G. Levy, B. Mazzolai, B. Hochner, C. Laschi, and P. Dario, "An octopus-bioinspired solution to movement and manipulation for soft robots," *Bioinsp. Biomim.*, vol. 6, p. 036002, 2011.
[11] T. Li, K. Nakajima, M. Kuba, T. Gutnick, B. Hochner, and R. Pfeifer, "From the Octopus to Soft Robots Control: an Octopus Inspired Behavior Control Architecture for Soft Robots," *Vie Milieu*, vol. 61, pp. 211-217, 2011.
[12] M. Lukosevicius and H. Jaeger, "Reservoir Computing Approaches to Recurrent Neural Network Training," *Comput. Sci. Rev.*, vol. 3, pp. 127-149, 2009.
[13] T. Li, K. Nakajima, M. Cianchetti, C. Laschi, and R. Pfeifer, "Behavior Switching by Using Reservoir Computing for a Soft Robotic Arm," in *Proceedings of the 2012 International Conference on Robotics and Automation(ICRA'12)*, pp. 4918-4924, 2012.
[14] J. Kuwabara, K. Nakajima, R. Kang, D. T. Branson, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, "Timing-Based Control via Echo State Network for Soft Robotic Arm," in *Proceedings of the 2012 International Joint Conference on Neural Networks (IJCNN)* (to appear), 2012.
[15] J. A. Mather, "How Do Octopuses Use Their Arms?," *J. Comp. Psychol.*, vol. 112, pp. 306-316, 1998.
[16] J. Z. Young, *The anatomy of the nervous system of Octopus vulgaris*. Oxford: Clarendon Press, 1971.
[17] M. J. Wells, *Octopus: physiology and behaviour of an advanced invertebrate*. London; New York: Chapman and Hall, 1978.
[18] R. A. Byrne, M. J. Kuba, D. V. Meisel, U. Griebel, and J. A. Mather, "Octopus arm choice is strongly influenced by eye use," *Behav. Brain. Res.*, vol. 172, pp. 195-201, 2006.
[19] R. A. Byrne, M. J. Kuba, D. V. Meisel, U. Griebel, and J. A. Mather, "Does Octopus vulgaris Have Preferred Arms?," *J. Comp. Psychol.*, vol. 120, pp. 198-204, 2006.
[20] Y. Gutfreund, H. Matzner, T. Flash, and B. Hochner, "Patterns of Motor Activity in the Isolated Nerve Cord of the Octopus Arm," *Biol. Bull.*, vol. 211, pp. 212-222, 2006.
[21] M. J. Wells, "Proprioception and Visual Discrimination of Orientation in Octopus," *J. Exp. Biol.*, vol. 37, no. 3, pp. 489-499, 1960
[22] M. J. Wells, "Tactile Discrimination of Surface Curvature and Shape by the Octopus," *J. Exp. Biol.*, vol. 41, no. 2, pp. 433-445, 1964.
[23] L. Zullo, G. Sumbre, C. Agnisola, T. Flash, and B. Hochner, "Nonsomatotopic Organization of the Higher Motor Centers in Octopus," *Curr. Biol.*, vol. 19, pp. 1632-1636, 2009.
[24] Y. Yekutieli, R. Sagiv-Zohar, B. Hochner, and T. Flash, "Dynamic Model of the Octopus Arm. II. Control of Reaching Movements," *J. Neurophysiol.*, vol. 94, pp. 1459-1468, 2005.
[25] G. Sumbre, Y. Gutfreund, G. Fiorito, T. Flash, and B. Hochner, "Control of octopus arm extension by a peripheral motor program," *Science*, vol. 293, pp. 1845-1848, 2001.
[26] J. Z. Young, "Multiple Matrices in the Memory System of Octopus", pp. 431-443, in *Cephalopod Neurobiology*, J. N. Abbott, R. Williamson, and L. Maddock, eds. Oxford University Press, Oxford 1995.
[27] M. Calisti, A. Arienti, F. Renda, G. Levy, B. Hochner, B. Mazzolai, P. Dario, and C. Laschi, "Design and Development of a Soft Robot with Crawling and Grasping Capabilities," in *Proceedings of the 2012 International Conference on Robotics and Automation(ICRA'12)*, pp. 4950-4955, 2012.
[28] E. Ackerman, "Robotic Octopus Takes First Betentacled Steps," http://spectrum.ieee.org/automaton/robotics/robotics-hardware/robotic-octopus-takes-first-betentacled-steps, accessed on the 15th April 2012.

Appendix F

Short-Term Memory in a Silicone-Based Soft Robotic Arm

Reprinted from:

Nakajima, K., Li, T., Hauser, H., Iida, F., and Pfeifer, R. (2013). Short-Term Memory in a Silicone-Based Soft Robotic Arm, submitted to *Nature Communications*.

The paper has been submitted and is still under review.

Short-term memory in soft body dynamics

Kohei Nakajima,[1*] Tao Li,[2] Helmut Hauser,[2] Fumiya Iida,[1] Rolf Pfeifer[2]

[1] *Department of Mechanical and Process Engineering, ETH Zurich, Leonhardstrasse 27, 8092 Zurich, Switzerland and*
[2] *Department of Informatics, University of Zurich, Andreasstrasse 15, 8050 Zurich, Switzerland.*

Soft materials are not only highly deformable, but they also possess rich and diverse body dynamics. Here we demonstrate that such soft body dynamics can be employed to conduct certain types of computation. Using body dynamics generated from a soft silicone arm, we show that they can be exploited to emulate functions that require memory and to embed robust closed-loop control into the arm. Our results suggest that soft body dynamics have a short-term memory and can serve as a computational resource. This finding paves the way towards exploiting softness for control of a large class of systems and a novel understanding of the notion of computation.

Recently, soft materials have been widely used to incorporate flexible elements into robots' bodies. The resulting machines, called *soft robots*, have significant advantages over traditional articulated robots in terms of morphological deformability and interactional safety [1]. They can adapt their morphology to unstructured environments, and carry and touch fragile objects without causing damage, which makes them especially applicable for rescue and human interactions, in particular care for the elderly, prosthetics, and wearables [2, 3]. In addition, they can generate diverse behaviors with simple types of actuation by partially outsourcing control to the morphological and material properties of their soft bodies [4], which is made possible by the tight coupling between control, body, and environment [5, 6]. In this paper, we build on these perspectives and add a novel advantage of soft bodies, demonstrating that they can be exploited as computational resources.

One of the major differences between rigid and soft bodies can be found in their body dynamics. Soft body dynamics usually exhibit a variety of properties including non-linearity, elasticity, and potentially infinitely many degrees of freedom, which are difficult to reduce to lower dimensionality and to control with conventional frameworks. We demonstrate that these properties can in fact be highly beneficial in that they can be employed for computation. Our approach is based on a machine learning technique, called *reservoir computing*, which is particularly suited to emulate complex temporal computations [7–11]. By driving a high dimensional dynamical system, typically referred to as the *reservoir*, with a low dimensional input stream, transient dynamics are generated that operate as a type of temporal and finite kernel [8, 11, 12]. If the dynamics involve enough non-linearity and memory, emulating complex, non-linear dynamical systems only requires adding a linear, static readout from the high-dimensional state space of the reservoir. A number of different implementations for reservoirs have been proposed: for example, abstract dynamical systems for echo state networks [7, 9], or models of neurons for liquid state machines [8]. Implementations even include using the surface of water in a laminar state [13]. Lately, it has been demonstrated that non-linear mass spring systems have the potential to serve as reservoirs as well [14, 15], and this has been applied in a number of ways (see, for example, [16, 17]).

In this study, we establish a simple but powerful physical platform with a soft silicone arm and demonstrate, through a number of experiments, that the soft body dynamics can be used as a reservoir. In particular, we focus on the property of short-term memory [18–20], which is the ability to store information about recent input sequences in the transient dynamics of the reservoir. In neuroscience, this property has drawn attention as a mechanism to perform real-time computations on sensory input streams, which is a prerequisite for cognitive phenomena, such as planning and decision making in the brain [11, 21, 22]. We show that short-term memory also exists in the body dynamics of a soft silicone arm and, in particular, that it can be exploited to control the arm's motions robustly in a closed-loop manner. In other words, the seemingly undesirable properties of soft body dynamics are no longer drawbacks for control but constitute core aspects of the system's functionality.

RESULTS

A soft silicone arm as a computational resource

There has been several soft silicone arms proposed in the literature, which are inspired by the octopus (see, for example, [23–25]). In this paper, we use a soft silicone arm, which has a similar material characteristic to the one proposed in [23]. The platform consists of a soft silicone arm, its sensing and actuation systems, data processing via a PC, and a water tank containing fresh water as an underwater environment (Fig. 1). By rotating the base of the arm and generating body dynamics induced by the interaction between the underwater environment

*nakajima@ifi.uzh.ch

and the soft silicone material, we aim to show that the sensory time series that reflects the body dynamics can be exploited as a part of a computational device. The unit of timestep t used in this study is a sensing and actuation loop of the PC. Throughout this study, we observe the behavior of the system from one side of the tank and use a terminology, such as "left" or "right," with respect to this point of view.

The arm embeds 10 bend sensors within a silicone material (Fig. 1a and Supplementary Fig. S1). A bend sensor gives a base value when it is straight. If it bends in the ventral side, the sensor value is smaller and if it bends dorsal the value is larger; the change in value reflects the degree of bend in each case. The sensors are embedded near the surface of the arm, with their ventral sides directed outward. We numbered these sensors from the base toward the tip as $s1$ through $s10$. The sensors are embedded alternately, with odd-numbered sensors ($s1$, $s3$, $s5$, $s7$, $s9$) on the right side of the arm and the even-numbered sensors ($s2$, $s4$, $s6$, $s8$, $s10$) on the left (Fig. 1a and Supplementary Fig. S1). The base of the arm can rotate left and right through the actuation of a servo motor. The motor commands sent from the PC are binary values, $M = \{0, 1\}$. If the command gives 0 or 1, then the motor is controlled to move from its current position toward the maximum right position (L_{right}) or the maximum left position (L_{left}), respectively (Fig. 1b). The actual servo motor positions are also sent to the PC to monitor the current position of the base rotation $\theta(t)$. The positions, L_{right} and L_{left}, were heuristically determined, in order to avoid damaging the motor components. The values for $|L_{right}|$ $(= |L_{left}|)$ are about 46.4 degrees by setting the origin of the rotation angle (0 degrees) when the arm is aligned vertically to the water surface. Throughout this study, $\theta(t)$ is linearly normalizes to be in the range from 0 to 1. Note that the motor command does not always take the roller position to L_{right} or L_{left}; rather, it decides the motor movement direction for each timestep. In addition, if the command gives 0 or 1, when the current position is L_{right} or L_{left}, respectively, then the position will stay unchanged.

To exploit the soft silicone arm as a computational resource, we need to determine how to provide inputs $I(t)$ to the system and how to generate corresponding outputs $O(t)$. In this paper, we provide the input to the motor command, $m(t) \in M$, and the output is generated by linearly combining all sensory time series $s_i(t)$ $(i = 1, 2, ..., 10)$ with a weighted sum using the weights w_i $(i = 1, 2, ..., 10)$. In addition, a bias is added, which is expressed as $b = w_0 s_0(t)$, where $s_0(t)$ is a constant value set to 1 (Fig. 2). As a result, we have 11 pairs of weights and corresponding sensory time series $(w_i, s_i(t))$ $(i = 0, 1, ..., 10)$ in our system. Our system output takes a binary state, $O(t) \in \{0, 1\}$, which is obtained by thresholding the weighted sum of the sensory values (see Methods section for details). To emulate a desired I/O function with our system, first, we apply the inputs to the system, which then generate the arm motions and we collect the corresponding sensory time series. Together with the target outputs, we have a training data set for supervised learning. The weights are then optimized with simple logistic regression with respect to minimize the error between the system output and the target output. The performance of the system output is evaluated by comparing with the target output for a new experimental trial (see Methods section and Supplementary Methods for detailed information).

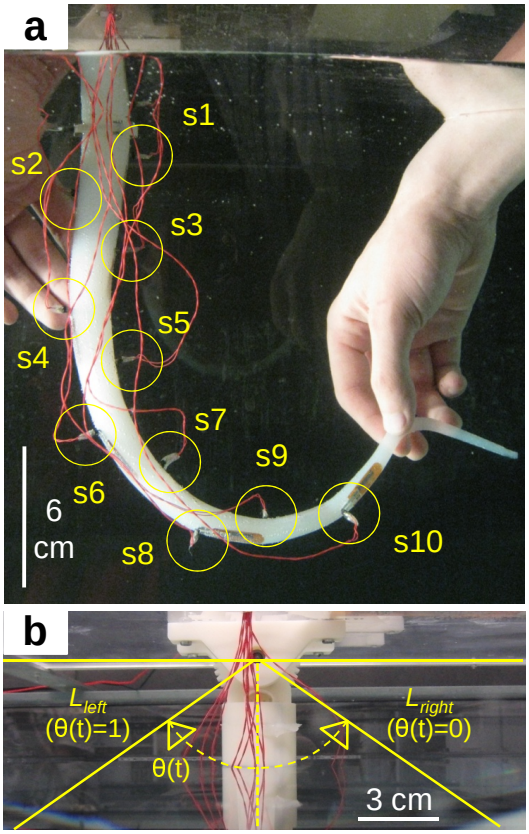


FIG. 1. **Platform setup for a soft silicone arm.** (a) A soft silicone arm, which contains 10 bend sensors. The arm is immersed in an underwater environment. Sensors are connected by red wires that send sensory values to a sensory board. The wires are set as carefully as possible so as not to affect the arm motion. (b) Motor commands take binary states, $m(t) \in M$. When these commands are set to 0 (1), the base of the arm rotates to the right (left) hand side toward L_{right} (L_{left}). See text for details.

We used three tasks to evaluate the computational power of our soft silicone arm, especially, with the focus on the property of short-term memory. Unlike the conventional computer, our system does not contain ex-

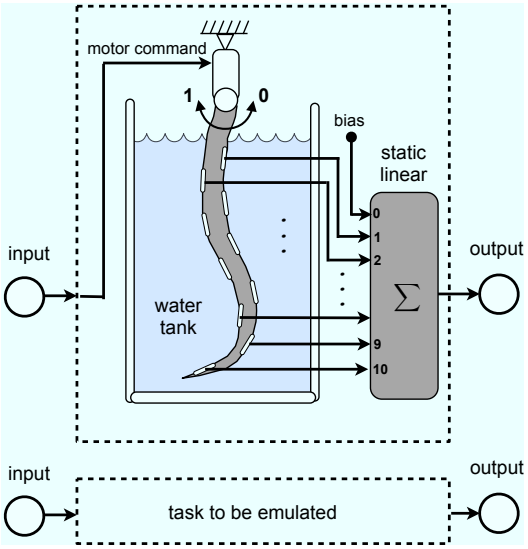


FIG. 2. **Schematics showing the information processing scheme using the arm.** Input is provided to the motor command to generate arm motion, and the embedded bend sensors reflect the resulting body dynamics. By using the detected sensory time series, the binary state output is generated by thresholding the weighted sum of the sensory values. Note that we include a bias term b, which is expressed as $b = w_0 s_0(t)$, where $s_0(t)$ is a constant value set to 1. Accordingly, we have 11 sensory values and corresponding weights. See text for details.

plicit memory storage; instead, the memory is expected to be implicitly included in the transient dynamics of the soft body. By assigning a task to the system that requires memory to be performed and by evaluating its task performance, we can characterize its hidden memory capacity. Our first task is to construct a timer exploiting the soft body dynamics. Triggered by a cue sent at certain timestep, the arm starts to move from L_{right} to L_{left}. We investigate how large of a pulse length the system is able to generate by exploiting the body dynamics. To perform this task, the system has to be able to "recognize" the duration of time that has passed since the cue was launched. This clearly requires memory.

The second task is to perform a closed-loop control exploiting the soft body dynamics. With a periodic square wave function, which switches its motor command from 0 to 1 and from 1 to 0 with a fixed period, as a target function, we aim to evaluate the maximal length of period for the square function the system can embed. In this task, the system should "recognize" how much time has passed since the motor command switched from 0 to 1 (or from 1 to 0), and it should decide when to switch the motor command to the next position. Again, this task requires memory. Furthermore, this task also evaluates whether the soft body dynamics can be exploited as a computational resource to control its own motion. This is especially interesting as typically the complex dynamics of a soft body are the main obstacles to apply classic control theoretic approach. Remarkably, in our proposed context, this property is beneficial as it can be exploited as a computational resource.

The third task is an emulation task of functions that requires memory. A random binary input sequence is provided to the system, and, by exploiting the generated soft body dynamics, the system should emulate two functions simultaneously; The first one is a function that reproduces past inputs with a given delay, and the second one is the N-bit parity checker. Again, both functions require memory. In particular, these functions should be emulated using the same soft body dynamics at the same time. This procedure is typically referred to as *multitasking* [14].

In all three tasks, we are only adjusting the static and linear weights, i.e., no memory is present in the readout. Hence, we can confirm that the required memory is purely due to the property of the soft silicone arm. Unlike conventional computational units, such as an artificial neural network, our proposed setup has a constraint due to the specifications of the mechanical structure of the system, since inputs are transformed to the mechanical realm. For example, a drastic and frequent switching of the motor command can result in motor overheat and a total stop. We defined the presented tasks to evaluate the memory capacity of our system by taking these physical constraints into consideration. Accordingly, as will be evident later, the I/O setting in our system slightly differs in each task (see also Methods section for details).

Basic property of the soft body dynamics

We start by investigating the dynamic property of our arm motion and the corresponding sensory time series. Figure 3a shows a typical arm motion when the motor command is switched from 0 to 1. The arm is initially set to L_{right}, and at $t = 0$, it starts to move towards L_{left}. As we can see in Fig. 3a, the soft robotic arm shows characteristic body dynamics because of the interaction between the water friction and the material properties of silicone (see also Supplementary Movie 1). In particular, even when the base reaches the position of L_{left}, the entire arm still shows transient dynamics. The figure clearly shows that because the arm moves from right to left, the right side of the arm bends and the left side of the arm arches according to the water friction.

The dynamic behavior of the arm can be captured by the responses of the sensors (Fig. 3b and Supplementary Movie 1). An important point to note here is that, when the motor command switches from 0 to 1, $\theta(t)$ takes about 9 timesteps to reach $\theta(t) = 1$, which forms a physical constraint based on the motor and the mechanical structure of our platform (Fig. 3b, upper figure, and

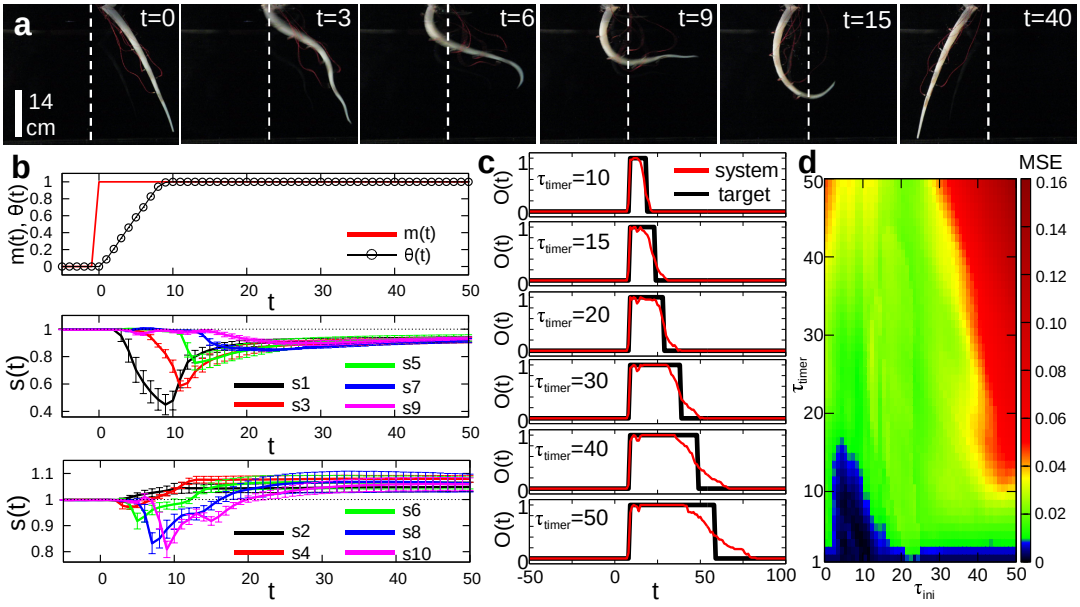


FIG. 3. **Sensory response during the arm motion and the timer task.** (a) Snapshots showing a typical arm motion when the motor command is switched from 0 to 1 (i.e., movement from L_{right} to L_{left}). (b) Plots showing the dynamics of the motor command $m(t)$ and the normalized base angle $\theta(t)$ (the upper plot) and the corresponding sensory time series $s(t)$ (the lower two plots). The motor command is switched from 0 to 1 at $t = 0$. The middle and lower plots show the average sensory response curves for the odd- and even-numbered sensors, respectively. For each sensor, the sensory values are linearly scaled to make the sensory values when the arm is in L_{right} to 1 and averaged over 50 trials. The error bars show the standard deviations. (c) The plots show the average system outputs for each τ_{timer} ($\tau_{timer} = 10, 15, 20, 30, 40$, and 50) when τ_{ini} is fixed to 9. The black lines show the target output and the red lines show the averaged system outputs over 25 trials for each condition. (d) The plot shows the average MSE over 25 trials with respect to each τ_{timer} and τ_{ini} varied from 1 to 50 and from 0 to 50, respectively.

Supplementary Movie 1). When the motor command is switched from 0 to 1, all the odd-numbered sensors start to show smaller values than those shown before the motion generation. They take a local minimum at different timestep, then gradually approaching their resting states (Fig. 3b, middle figure, and Supplementary Movie 1). According to the plots, the movement of the base rotation propagates from the base toward the tip at a certain velocity. For example, $s1$ seems to show a direct reflection of the motor actuation, because it is embedded close to the base. This effect can be confirmed by checking the local minimum of the sensory response of $s1$ at around timestep 9, which is the same timestep at which the motor rotation stops. For even-numbered sensors, although all sensors show larger values than the values before the motion generation, some sensors, such as $s6$, $s8$, and $s10$, show a smaller value in some timesteps due to inertia caused by the immediate bend in the left side of the arm (Fig. 3b, lower figure, and Supplementary Movie 1). This effect also seems to be propagating from the base toward the tip of the arm. All sensors reach a resting state at around 40 timesteps. Note that in the L_{left} resting state the odd-numbered sensors show smaller values and the even-numbered sensors show greater values than those shown before motion generation (Fig. 3b and Supplementary Movie 1). This phenomenon is the result of gravity; the left side of the arm arches slightly, while the right side of the arm bends slightly (Fig. 3a). When the motor command is switched from 1 to 0 with the arm position initially set to L_{left}, we can confirm similar tendencies, now with switched roles of the odd- and even-numbered sensors.

Timer task

Our first task is to emulate the function of a timer exploiting the body dynamics of the arm. The task has been chosen as it enables us to investigate systematically the memory inherently present in the soft body dynamics. One of the characteristic properties of our soft body is its transient dynamics during its motion from one state to another, e.g., move from left to right. In this task, the arm is initially set to L_{right} and kept at

this position. Triggered by the input at t_{start}, the motor command switches from 0 to 1, when the rotation of the base generates the body dynamics (Fig. 3a). The timer task consists of producing an output pulse starting from τ_{ini} timesteps after t_{start}, which is τ_{timer} timesteps in length, by exploiting the body dynamics during this transient single motion (Supplementary Figure S2, see Methods section for details). In order to perform this task, the system has to have a certain amount of memory. In other words, we can evaluate whether the sensory time series that reflects the transient dynamics during the motion from L_{right} to L_{left} contains sufficient information to recognize the duration of time by applying this task. A similar task was introduced in [7] to demonstrate the existence of fading memory within an artificial recurrent neural network called echo state network [9, 18]. In order to demonstrate that such a memory can be found and exploited in a real physical system (if the body is soft enough), we applied this task employing the soft silicone arm. As explained earlier, our system output is generated by thresholding the weighted sum of the sensory values, and the weights are optimized with a simple logistic regression by using a data set collected in the training phase (see Methods section and Supplementary Methods for detailed information). We performed this experiment by varying τ_{ini} and τ_{timer} to investigate the relevance of these parameters to the system performance.

Figure 3c shows examples of the averaged system outputs for each τ_{timer} with τ_{ini} fixed to 9. As one can see, our system is able to emulate a timer with given duration times τ_{timer} (see also Supplementary Movie 1); naturally, the performance decreases when increasing the length of τ_{timer}. This is caused by the gradual fading of memory within the body dynamics after the initiation of motion generation. This tendency can be found for different setting of τ_{ini} (Fig. 3d). As can be seen in Fig. 3d, the error values (MSE) are especially low when around $\tau_{ini} < 20$ and $\tau_{timer} < 20$, characterizing the amount of memory that can be exploited with the given soft body. Note that when τ_{ini} is close to 0, the error values are higher than for other parameters. This is because, when the arm starts to move, the effect of the motor rotation takes some time to propagate due to the softness of the arm (Figs. 3a and 3b), and if τ_{ini} is small, it is difficult to distinguish the sensory values from the values when the arm is stopped.

Closed-loop control task

We demonstrated in the previous task that we can use the sensory time series generated by the transient dynamics to construct a timer. By using the same property, in this second task, we aim to realize a closed-loop control of our soft silicone arm. That is, we aim to demonstrate that the arm's body dynamics can be used to control its own motion. The target motor command sequence is a square wave in which the amplitude alternates at a steady frequency, between $m(t) = 0$ and 1, with the same duration of timesteps, τ_{square} (Supplementary Fig. S3a; see Methods section for details). Similar to the process in the previous task, when the motor command is switched from 0 to 1 (or from 1 to 0), it should recognize the time length of τ_{square} timesteps and switch the motor command from 1 to 0 (or from 0 to 1). Thus, it is required to have memory to fulfill this task. We aim to emulate this oscillatory wave pattern by using the sensory time series. This is realized by feeding back the system output generated by thresholding the weighted sum of the sensory time series as the next motor command to the system (Supplementary Fig. S3b; see Methods section for details). As with the previous task, we aim to emulate the target output only by adjusting the static linear weights (see Methods section for details on the experimental procedure).

Figure 4a shows an example of a time series with the motor commands and sensory values when the system is driven by the closed-loop control emulating a square function with $\tau_{square} = 10$. We can see that the time series of the motor command exactly overlaps with the target output, showing that the closed-loop control is successfully embedded (see Supplementary Movie 2 for the arm motion). For real-world applications, it is important to investigate whether the system is robust against external perturbation. We investigated the robustness of the system by applying a mechanical perturbation and disturbing the arm motion by hand (Figs. 4b and 4c). We found that, during the perturbation, both the sensory time series and the system output were affected; however, after removing the disturbance, the system immediately recovered its original trajectory (see Supplementary Movie 2 for the arm motion during the perturbation). This can be confirmed by checking the time series of the motor commands and their corresponding sensory values, and implies that our system is robust against external perturbations (Fig. 4b). Note that, although the system output shows a phase shift compared with the target output after the perturbation, it is generating a square function with a required length of τ_{square}.

To evaluate the maximal length of τ_{square} of a square function our system can embed, we investigated an average system output for one period of a square function by clamping the feedback loop from the system output and providing the target output as input for each τ_{square} (Fig. 4d, see Methods section for details). There is a reason to clamp the feedback loop: if the system is driven by the closed-loop control, the error in the system output would propagate to the motor command through the feedback loop, which makes it difficult to evaluate the limitation of the system performance efficiently. As can be seen from the plot, according to the increase of τ_{square}, the average system output starts to deviate largely from the target output. By calculating the system error by means of MSE in this setting, we found that the error grows immediately when τ_{square} gets larger than 18 (Fig. 4e, see Methods section for details). Consistent with this result, we observed that when τ_{square} is more than 18, the

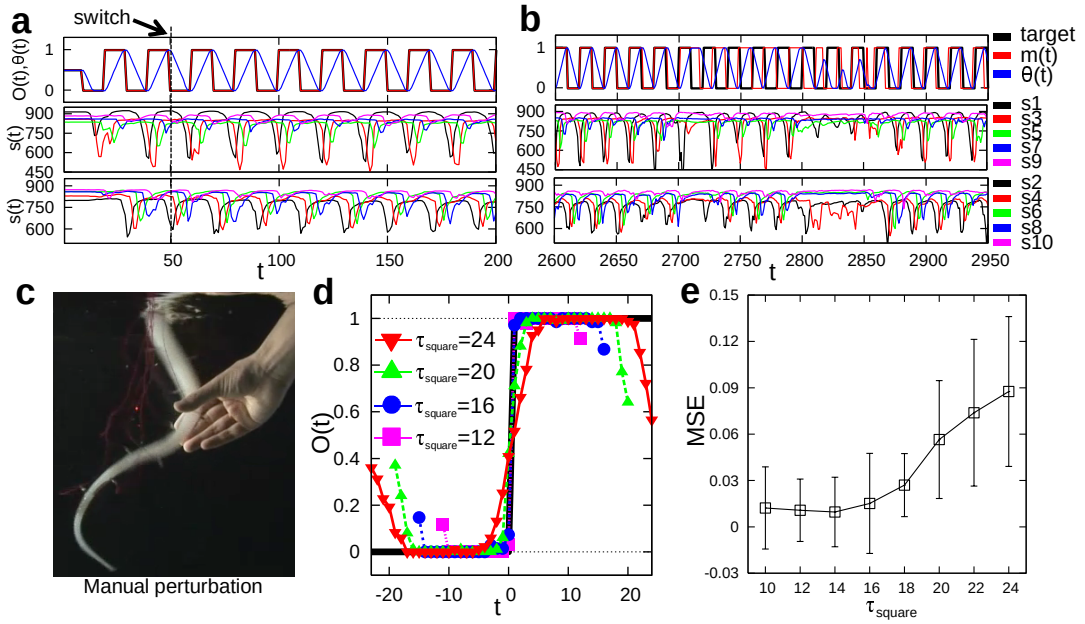


FIG. 4. **Performance of the closed-loop control task.** (a) Plots showing an example of the sensorimotor time series when the system is driven by the closed-loop control with a square function of $\tau_{square} = 10$. The system is initially driven by the target output till $t = 50$ and then switched to the system output. The upper diagram plots the time series of the target output, the system output, which is the motor command $m(t)$ in this task, and the base rotation $\theta(t)$. The middle and lower diagrams plot the corresponding sensory time series $s(t)$ in odd and even numbers, respectively. (b) Plots showing an example of the sensorimotor time series when the system is driven by the closed-loop control with the external perturbations in the same experimental condition with (a). The perturbation to the arm is provided from around 2700 timestep to 2850 timestep. (c) The external perturbation is provided by the manual mechanical disturbance to the arm. (d) The average system outputs for a single period of a square function driven by clamping the feedback loop from the system output and providing the target output as input. The plots for $\tau_{square} = 12, 16, 20$, and 24 are overlaid. The black line shows the target values as a reference. The time is shifted to set the switching point timestep at 0. (e) The average error (MSE) plot with respect to each τ_{square}. The error bars show the standard deviations. See Methods section for detailed information on the analysis.

system cannot embed a correct square function anymore, or it simply stops, continuously providing 0 or 1 as output. This happens especially when externally perturbed (see also Supplementary Movie 2). Thus, we can speculate that our system possesses enough memory to be exploited for embedding a square function up to a length of around $\tau_{square} = 18$.

Function emulation tasks: short-term memory and N-bit parity checker

In this final task, we aim to quantitatively characterize the intrinsic computational capacity of our system, particularly focusing on its memory capacity. By providing a random binary sequence to the motor command as input, the system should perform function emulation tasks by using the resulting sensory time series. As explained earlier, because our system is not an abstract computational unit, but has physical and mechanical constraints, we need to define a certain duration of time for one input state or symbol. We call this duration of time τ_{state}. We found that when we provide a random binary sequence as motor commands in the form of $\tau_{state} < 5$, the motor stops due to overheating. Accordingly, we performed our experiments with $\tau_{state} \geq 5$. In addition, because of this setting, we introduced a different time scale for I/O, defined as t', which takes one input symbol as a unit. This means that t' is incremented by 1 for each τ_{state} timesteps (Supplementary Fig. S4a; see Supplementary Methods for details).

The first function that we aim to emulate is a function that provides delayed version of the input, i.e., $I(t' - n)$ $(n = 1, 2, ...)$ (see Methods section for details). This task enables us to directly evaluate whether the system contains memory traces of a past input within the current state of the system or not. This task has been introduced in [18] and is frequently used to evaluate the memory

capacity of systems (see, for example, [19, 21]). For descriptive purpose, we call this task the *short-term memory task*. The second function we aim to emulate is the N-bit parity checker. The output should provide 0 if $\sum_{d=0}^{n} I(t'-d)$ is an even number; otherwise, should provide 1, with $n = 1, 2, ...$ (see Methods section for details). (Note that it is actually "$(n+1)$-bit parity checker" in our case.) According to the definition, the system needs the memory of input symbols to n symbols before within the system to emulate this function. In addition, this function is a non-linear function for $n+1$ input symbols [22]. Thus, this task allows us to evaluate whether the system contains sufficient memory and non-linearity to be exploited. Along with the definition of the input symbol in this experiment, we also need to determine how to define a corresponding sensory time series. Let us assume that an input symbol was provided at timestep $t(= t'\tau_{state})$. As a result, the arm generates corresponding transient dynamics until the next input symbol is provided at timestep $(t'+1)\tau_{state}$. We define sensory values at $(t'+1)\tau_{state}-1$ as corresponding values, $s_i(t')$, for this input symbol, which is one timestep before the next input symbol is provided (Supplementary Fig. S4a; see Methods section for details). By providing random binary input sequences to the system over several trials for each parameter τ_{state} and n, we collected the sensory time series used for training. Both target functions are simultaneously emulated over one random input sequence (Supplementary Fig. S4b; see Methods section for details).

Figures 5a and b show examples of the system performance for the short-term memory task with $\tau_{state} = 5$ and the N-bit parity check task with $\tau_{state} = 11$ (see also the arm motion in Supplementary Movie 3). We can see that the system output shows a perfect match with the target output when $n = 1$ and 2 for the short-term memory task and when $n = 1$ for the N-bit parity check task. For both tasks, the performance gradually gets worse when n is increased. To evaluate the influence of the parameters of τ_{state} and n on the system performance, we introduce measures based on mutual information, MI_n, between the system output and the target output [20]. This measure evaluates the similarity between the system output and the target output and can take the value of 1 as maximum and 0 as minimum in our experiment (see Methods section for details). Additionally, we introduce a measure called "capacity," which is a summation of MI_n over the delays, expressed as $C = \sum_{n=1}^{n_{max}} MI_n$, where n_{max} is set to 10 in this analysis (see Methods section for details).

Figures 6a and b show the results of the average MI_n for each n value and the average capacity for each τ_{state} for each task (see Methods section for details). For the short-term memory task, we can see that, when τ_{state} is increased, the value of MI_n suddenly drops when n is larger than 2 (Fig. 6a, left diagram). For the capacity, increasing τ_{state} results, first, in a gradual decrease and then in saturation at the constant value for $\tau_{state} > 11$

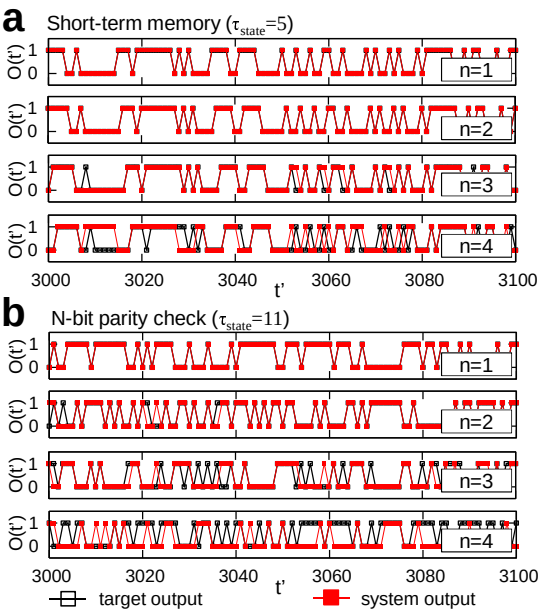


FIG. 5. **Examples of the output time series for the function emulation tasks.** (a) Plots showing the example of the performance in the short-term memory task with $\tau_{state} = 5$. (b) Plots showing the example of the performance in the N-bit parity check task with $\tau_{state} = 11$. For (a) and (b), the black line shows the target outputs and the red line shows the system outputs, and the cases for $n = 1, 2, 3$, and 4 are shown.

(Fig. 6a, right diagram). This can be explained by the behavior of the arm (see also Supplementary Movie 3) – if the length of the input symbol is short, it is more likely that the current transient dynamics contains the trace of previous input symbols provided. Considering that the arm base takes about 9 timesteps to get from one end to the other, if τ_{state} gets larger than 9 timesteps, the arm can more or less possess the information about the last input symbol, because of the simple one-way bend motion. This explains the maximal performance with respect to MI_n when $n = 1$. In order to see the contribution of the physical body to the computational task, we compared the performance with a model that has a readout directly attached to the input (see Methods section for details). We can confirm that this model cannot perform this task at all, which suggests that the performance of our system is purely based on the body dynamics.

For the N-bit parity check task, we can see that, even if τ_{state} is small ($\tau_{state} = 5$), when n = 1 (Fig. 6b, left diagram), MI_n shows a smaller value than when τ_{state} is larger ($\tau_{state} = 10$ and 20). When τ_{state} gets larger ($\tau_{state} = 10$), MI_n starts to show the highest value when $n = 1$, and a moderately high value in $n = 2$. If we

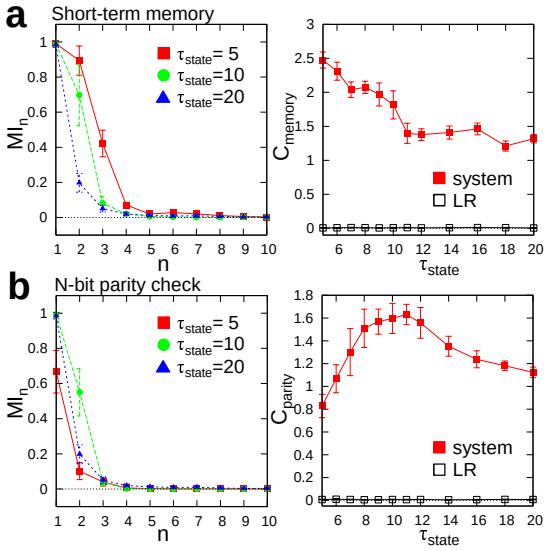


FIG. 6. **The average value of MI_n according to n (left figure) and C according to τ_{state} (right figure).** (a) Plots showing the case for the short-term memory task. (b) Plots showing the case for the N-bit parity check task. Note that the capacities in the short-term memory task and the N-bit parity check task are expressed as C_{memory} and C_{parity}, respectively. For each plot on MI_n, the cases for $\tau = 5, 10$, and 20 are shown. For each plot on C, the results of a logistic regression model (LR) that has a readout directly attached to the input (see Methods section for details) are also plotted as comparisons, and they show almost 0 value for each τ_{state}. The error bars show the standard deviation for each plot.

increase τ_{state} further ($\tau_{state} = 20$), MI_n still shows the highest value when $n = 1$, but the value in $n = 2$ starts to decrease. This tendency reflects the results of the capacity (Fig. 6b, right diagram). As one can see, the capacity shows a peak around $\tau_{state} = 9, 10$, and 11. The low values of capacity in τ_{state} less than 9 and larger than 11 are because of the low values of MI_n in $n = 1$ and $n = 2$, respectively. In this task also, the model with a readout directly attached to the input cannot perform the emulation at all. Considering that the N-bit parity check task requires not only memory but also non-linearity to perform, this result suggests that, even if the transient dynamics of the arm possess a high memory capacity when τ_{state} is low, it does not contain sufficient non-linearity to be exploited. This is interesting, since this result is not visually detectable by just simply looking at the arm motion. Furthermore, the results show that the amount of computational capacity depends on the type of motion generated in the arm.

CONCLUSION AND DISCUSSION

In this study, we have demonstrated that the body dynamics of the soft silicone arm can be exploited as a computational resource. In particular, for the closed-loop control task, our results suggest that the soft body dynamics are only sufficient to perform the control task, without any support from external controller for additional memory capacity. The technique presented in this study can be potentially applied to a wide class of soft robots because the main component required is a soft body itself, which is already present in soft robotic platform. Consequently, different types of morphology and material properties of robots, which increase the computational capacity of the body, should be explored in the future. In addition, developments on new types of sensors, which can effectively monitor body dynamics, would make the presented technique applicable for further applications. To conclude, we believe that we have presented crucial step toward a novel control scheme for soft robots.

When we turn our eyes to nature, biological systems have soft bodies, which can adapt and behave effectively in a given ecological niche. For example, the octopus does not have any rigid components in its body, but it shows extremely sophisticated behavior that capitalizes on its body morphology and muscle structures [27]. The framework presented in this study may also shed light on investigating the role of the body in biological systems. In particular, we have shown that a form of short-term memory, which is thought to be a functionality of the brain, can also be found in soft body dynamics. This line of studies is an interesting research direction to be explored further.

METHODS

Experimental platform setup

The experimental platform mainly consists of a soft silicone arm, its actuation, sensing and control systems, and a water tank containing fresh water as the working environment. The size of the water tank is 100 cm long, 50 cm wide, and 50 cm deep. During experiments, the arm is immersed in the water and actuated by a servo motor at the arm base, which consists of rigid plastic and is directly connected to the motor. For each experimental trial, the amount of water in the tank is controlled so it is the same height of the apical surface of the plastic material of the base when the arm is aligned vertically to the water surface. Sensors embedded in the arm are used to detect the amount of bending of the arm during experiments. The motor commands and sensory data are recorded at each control timestep for further analysis.

A soft silicone arm

We made the soft arm with silicone rubber (ECOFLEX^{TM}00-30 from Smooth-On Inc.) using an ABS plastic mold manufactured by a 3D printer (Fig. 1a and Supplementary Fig. S1a). The mold has two separate pieces and can be assembled together. The silicone arm has a corn shape. It is 44.7 cm long, has a radius of 1.4 cm at one end and 0.15 cm at the other end (Supplementary Fig. S1b), which is determined so as not to touch the ground and the walls during movement. Ten bend sensors were embedded near the surface of the silicone arm during the process of making the arm. There are four steps involved in making a silicone arm with embedded bend sensors: (1) align five bend sensors at the bottom of each piece of the mold; (2) pour a layer of silicone on the bend sensors so that the sensors' arrangement is fixed after the silicone is cured; (3) assemble the two separate pieces of the mold together and fill the remaining space in the mold with silicone; (4) open the mold and take out the silicone arm after the silicone is cured.

Bend sensors

To detect the body dynamics of the soft silicone arm, we used flexible lightweight bend sensors from Flexpoint Sensor Systems, Inc. (Supplementary Fig. S1c). The size of the sensor is roughly 3.2 cm long, including connectors, 0.7 cm wide, and less than 0.1 cm thick. It consists of a thin plastic base film, a layer of coated bend sensitive ink, and two connectors [28]. The coated bend sensitive ink layer changes its electrical conductivity, as the sensor is subjected to bending. Therefore, the sensor is actually a potentiometer, which converts mechanical deformation into the change of electric resistance. An advantage of the sensor is its large range of resistance change from a few hundred Ω to a few hundred $K\Omega$; thus, a simple voltage divider can be used to read the sensory output [29]. Typical sensor response curves can be found in the design manual of the provider [29].

Sensory data acquisition system

We used a sensor board with voltage dividers and a 16-channel multiplexer, an Arduino^{TM}MEGA 2560 board, and a PC for data acquisition. The fixed resistors of the voltage divider are 10 $K\Omega$. The multiplexer reads in data from each of its input channels serially and sends it to the Arduino board's analog input pins. Then the Arduino board transmits the bend sensors data to a PC serial port. Finally, a Java program running in the PC reads and records the sensory data at each timestep. Arranging the electrical cables connecting the sensor connectors and the sensor board is a challenge. Because of the repeated bending during experiments, the cables are prone to breakage, especially near the sensor connectors. Using an L-shape cable connector and putting the electrical cable completely outside of the arm, the cables are not only easy to change but also become free from bending stress.

Actuation system

The soft silicone arm is actuated by a Dynamixel RX-64 servo motor, which is controlled by the Java program running on the PC. The servo motor is fixed on a plexiglass plate put on top of the water tank. The servo motor rotation is transmitted to the soft silicone arm by two identical plastic gears: one is fixed at the end of the motor axle; the other is attached at the larger end of the silicone arm. Motor commands used for each experiment are described in the main text.

Experimental procedure for the timer task

As explained in the main text, our first task was to emulate the function of a timer exploiting the body dynamics of the arm. The I/O relation for the timer can be expressed as follows (Supplementary Fig. S2a):

$$I(t) = \begin{cases} 1 & (t = t_{start}) \\ 0 & (\text{otherwise}) \end{cases}$$

$$O_{target}(t) = \begin{cases} 1 & (t_{start} + \tau_{ini} \leq t \leq t_{start} + \tau_{ini} + \tau_{timer}) \\ 0 & (\text{otherwise}). \end{cases}$$

The behavior of the motor commands, $m(t)$, according to the input, $I(t)$, can be expressed as follows:

$$\begin{aligned} m(t) &= f(I(t)), \\ &= \begin{cases} 1 & (t \geq t_{start}) \\ 0 & (\text{otherwise}). \end{cases} \end{aligned}$$

Our aim was to emulate this timer by exploiting the body dynamics generated by the input. Our system outputs are produced by applying static linear readout weights w_i $(i = 0, 1, ..., 10)$ to the sensory time series $s_i(t)$ $(i = 0, 1, ..., 10)$ as follows (Supplementary Fig. S2b):

$$O_{system}(t) = P(\sum_{i=0}^{10} w_i s_i(t)),$$

$$P(x) = \begin{cases} 1 & (x > 0) \\ 0 & (\text{otherwise}), \end{cases}$$

where the system output $O_{system}(t)$ is obtained by thresholding function $P(x)$. In this task, the arm is initially set to L_{right} and the motor command is set to 0 $(m(t) = 0)$. At timestep 50, the input provides 1, and triggered by this input command, the motor command switches from 0 to 1 (that is, $t_{start} = 50$). After that, the system continues to run for another 200 timesteps. We

consider the overall 250 timesteps as one trial in this task. For the training procedure, we iterated this process over 25 trials and collected the corresponding sensory time series for each timestep. We optimized the linear output weights using these collected sensory time series with a logistic regression to emulate the target output for given τ_{ini} and τ_{timer} [26] (see Supplementary Methods for details). To evaluate the performance of the system with the optimized weights, we ran 25 additional trials (evaluation trials) and compared the system outputs to the target outputs. The average system output in Fig. 3c in the main text is obtained by averaging the system output using the evaluation trials for each timestep. In Fig. 3c, the time is shifted so that $t_{start} = 0$ for clarity. In addition, the mean squared error (MSE) in Fig. 3d is calculated as follows:

$$MSE = \frac{1}{T}\sum_{t=0}^{T}(O_{target}(t) - O_{system}(t))^2,$$

where T is 250 in this task. For each pair of parameters $(\tau_{ini}, \tau_{timer})$, the readout is trained in the above mentioned manner, and by using the evaluation trials, the average MSE is calculated.

Experimental procedure for the closed-loop control task

In this task, we aimed to embed a square function in a closed-loop. The used square wave function can be expressed as follows (Supplementary Fig. S3a):

$$x(t) = \frac{1}{2}(sgn(\sin(\frac{2\pi}{\tau_{square}}t)) + 1).$$

To emulate this oscillatory wave pattern by using the sensory time series, the generated output value is fed back as the next motor command to the system and is expressed as follows (Supplementary Figs. S3a and S3b):

$$\begin{aligned} m(t) &= I(t), \\ O_{system}(t) &= P(\sum_{i=0}^{10} w_i s_i(t)), \\ I(t+1) &= O_{system}(t). \end{aligned}$$

As for the timer task, we emulated the above square wave function only by adjusting the static linear output weights. Because this task requires feedback to the system, the training procedure is different from the previous one. During the training phase, we clamped the feedback from the system output, and provided the target outputs as inputs ($x(t)$ ($= O_{target}(t)$)), which means we set $I(t+1) = x(t)$. Thus, the training phase was carried out with an open-loop, such that the system was forced into the desired operative state by the target signals in the required τ_{square} (this approach is typically referred to as *teacher forcing*) [15]. In the experimental procedure, the soft silicone arm was first set to the resting state for τ_{square} timesteps to align the arm vertically to the water surface without motion. We first ran the system with the teacher forcing condition and collected the corresponding sensory time series data for 2500 timesteps. The first 100 timesteps were discarded, and the remaining 2400 timesteps were used for training. After obtaining the optimal readout weights from these collected data by the use of logistic regression, we initialized the system to the resting state and ran again the system with the teacher forcing condition. After 50 timesteps, we switched the inputs to the system output generated by the trained readout weights and checked whether the system was able to embed the square wave function robustly.

As is explained in the main text, to evaluate the performance of the system for each τ_{square}, we ran the system with the teacher forcing condition and collected the sensory time series for 50 cycles of motor oscillations and trained the weights as previously described. Then, by using the optimized weights, we ran a new trial of 50 cycles of motor oscillations as an evaluation phase with the teacher forcing condition and compared the system outputs and the target outputs. Figure 4d in the main text shows the averaged system output for this evaluation phase. For the figure, the time is shifted so that the time when the target output switches from 0 to 1 is at $t = 0$ for clarity. The plot in Fig. 4e in the main text is obtained by averaging the MSE (with $T = 2\tau_{square}$) in this evaluation phase.

Experimental procedure for the function emulation tasks

In this task, we aimed to emulate the short-term memory task and the N-bit parity check task. Following the notation defined in the main text, the function for the short-term memory task can be expressed as follows (Supplementary Fig. S4a):

$$O_n^{short}(t') = I(t' - n),$$

where $I(t')$ is a random binary sequence. The function for the N-bit parity check task is expressed as follows (Supplementary Fig. S4a):

$$\begin{aligned} O_n^{parity}(t') &= Q(\sum_{d=0}^{n} I(t' - d)), \\ Q(x) &= \begin{cases} 0 & (x \equiv 0 (mod 2)) \\ 1 & (\text{otherwise}) \end{cases} \end{aligned}$$

Here, the system output is generated as (Supplementary Fig. S4b):

$$O_{system}(t') = P(\sum_{i=0}^{10} w_i s_i(t')).$$

In this task, one trial consists of 3500 symbols (i.e., 3500*τ_{state} timesteps). The first 100 symbols are discarded, the next 2400 symbols are used for training, and the last 1000 symbols are used for the system evaluation. Both functions are emulated simultaneously using the same random input sequence. This is typically referred to as *multitasking* (Supplementary Fig. S4b). The training of the static linear readout weights is conducted by a logistic regression for each function emulation task. For each τ_{state}, we ran the system for 15 trials and evaluated the system performance with the target output for each n by using the mutual information expressed as follows [20]:

$$MI_n = \sum_{O_{system}(t)\in O} \sum_{O_{target}(t)\in O} p(O_{system}(t), O_{target}(t)) \log \frac{p(O_{system}(t), O_{target}(t))}{p(O_{system}(t))p(O_{target}(t))},$$

where $O = \{0, 1\}$, and $p(x)$ and $p(x, y)$ are the probability of x and the joint probability of x and y, respectively. The base of *log* is fixed to 2 throughout this study. When $O_{system}(t)$ and $O_{target}(t)$ are independent, then $MI_n = 0$. When $O_{system}(t)$ is exactly the same as $O_{target}(t)$, then $MI_n = 1$, since $O_{target}(t)$ is random. In Fig. 6, MI_n and the capacity C are averaged over 15 trials in each condition. In the main text, to see the contribution of the physical body to the computational task, we compared the performance with a logistic regression model that has a readout directly attached to the input, expressed as:

$$O_{LR}(t') = P(w_1 I(t') + w_0),$$

where the weights are trained by a logistic regression using the same time series as in the training phase for each function emulation task.

[1] Pfeifer, R., Lungarella, M., Iida, F. The challenges ahead for bio-inspired 'soft' robotics. *Commun. ACM.* **55**, 76-87 (2012).
[2] Trivedi, D., Rahn, C. D., Kier, W. M., Walker, I. D. Soft robotics: biological inspiration, state of the art, and future research. *Appl. Bionics. Biomech.* **5**, 99-117 (2008).
[3] Kim, S., Laschi, C., Trimmer, B. Soft robotics: a new perspective in robot evolution. *Trends Biotechnol.* **31**, 287-294 (2013).
[4] Shepherd, R. F., Ilievski, F., Choi, W., Morin, S. A., Stokes, A. A., Mazzeo, A. D., Chen, X., Wang, M., and Whitesides, G. M. Multi-gait soft robot. *Proc. Natl. Acad. Sci. U. S. A.* **108**, 20400-20403 (2011).
[5] Pfeifer, R., Lungarella, M., Iida, F. Self-organization, embodiment, and biologically inspired robotics. *Science* **318**, 1088-1093 (2007).
[6] Pfeifer, R., Bongard, J. *How the Body Shapes the Way We Think: A New View of Intelligence* (MIT Press, Cambridge, MA, 2006).
[7] Jaeger, H. Tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach. GMD Report 159, German National Research Center for Information Technology (2002).
[8] Maass, W., Natschläger, T., Markram, H. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput.* **14**, 2531-2560 (2002).
[9] Jaeger, H., Haas, H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* **304**, 78-80 (2004).
[10] Verstraeten, D., Schrauwen, B., D'Haene, M., Stroobandt, D. An experimental unification of reservoir computing methods. *Neural Netw.* **20**, 391-403 (2007).
[11] Buonomano, D.V., Maass, W. State-dependent computations: spatiotemporal processing in cortical networks. *Nat. Rev. Neurosci.* **10**, 113-125 (2009).
[12] Rabinovich, M., Huerta, R., Laurent, G. Transient dynamics for neural processing. *Science* **321**, 48-50 (2008).
[13] Fernando, C., Sojakka, S. Pattern recognition in a bucket. In *Lecture Notes in Computer Science* **2801**, 588-597 (Springer, 2003).
[14] Hauser, H., Ijspeert, A. J., Füchslin, R. M., Pfeifer, R., Maass, W. Towards a theoretical foundation for morphological computation with compliant bodies. *Biol. Cybern.* **105**, 355-370 (2011).
[15] Hauser, H., Ijspeert, A. J., Füchslin, R. M., Pfeifer, R., Maass, W. The role of feedback in morphological computation with compliant bodies. *Biol. Cybern.* **106**, 1-12 (2012).
[16] Nakajima, K., Hauser, H., Kang, R., Guglielmino, E., Caldwell, D. G., Pfeifer, R. A soft body as a reservoir: case studies in a dynamic model of octopus-inspired soft robotic arm. *Front. Comput. Neurosci.* **7**, 1-19 (2013).
[17] Caluwaerts, K., D'Haene, M., Verstraeten, D., Schrauwen, B. Locomotion without a brain: physical reservoir computing in tensegrity structures. *Artif. Life* **19**, 35-66 (2013).
[18] Jaeger, H. Short term memory in echo state networks. GMD Report 152, German National Research Center for Information Technology (2001).
[19] White, O., Lee, D., Sompolinsky, H. Short-term memory in orthogonal neural networks. *Phys. Rev. Lett.* **92**, 148102 (2002).
[20] Bertschinger, N., Natschläger, T. Real-time computation at the edge of chaos in recurrent neural networks. *Neural Comput.* **16**, 1413-1436 (2004).
[21] Ganguli, S., Huh, D., Sompolinsky, H. Memory traces in dynamical systems. *Proc. Natl. Acad. Sci. U. S. A.* **105**, 18970-18975 (2008).
[22] Nikolić, D., Häusler, S., Singer, W., Maass, W. Distributed fading memory for stimulus properties in the primary visual cortex. *PLoS Biol.* **7**, 1-19 (2009).
[23] Cianchetti, M., Arienti, A., Follador, M., Mazzolai, B., Dario, P., Laschi, C. Design concept and validation of a robotic arm inspired by the octopus. *Mater. Sci. Eng., C* **31**, 1230-1239 (2011).

[24] Calisti, M., Giorelli, M., Levy, G., Mazzolai, B., Hochner, B., Laschi, C., Dario, P. An octopus-bioinspired solution to movement and manipulation for soft robots. *Bioinsp. Biomim.* **6**, 036002 (2011).
[25] Martinez, R. V., Branch, J. L., Fish, C. R., Jin, L., Shepherd, R. F., Nunes, R. M. D., Suo, Z., Whitesides, G. M. Robotic tentacles with three-dimensional mobility based on flexible elastomers. *Adv. Mater.* **25**, 205-212 (2013).
[26] Bishop, C. M. *Pattern Recognition and Machine Learning* (Springer, 2006).
[27] Hochner, B. An embodied view of octopus neurobiology. *Curr. Biol.* **22**, R887-R892 (2012).
[28] Flexpoint Sensor Systems, Inc., Mechanical Design Guide, www.flexpoint.com/technicalDataSheets/mechanicalDesignGuide.pdf.
[29] Flexpoint Sensor Systems, Inc., Electronic Design Guide, www.flexpoint.com/technicalDataSheets/electronicDesignGuide.pdf.

ACKNOWLEDGMENTS

This work was partially supported by the European Commission in the ICT-FET OCTOPUS Integrating Project (EU project FP7-231608), and by JSPS Postdoctoral Fellowships for Research Abroad.

AUTHOR CONTRIBUTION

K.N., T.L., H.H. and R.P. developed the concept and designed experiments. K.N. and T.L. prepared the experimental setup and carried out experiments. K.N. and H.H. analyzed and interpreted the data. K.N., T.L., H.H., F.I. and R.P. discussed the results and implications and commented on the manuscript at all stages.

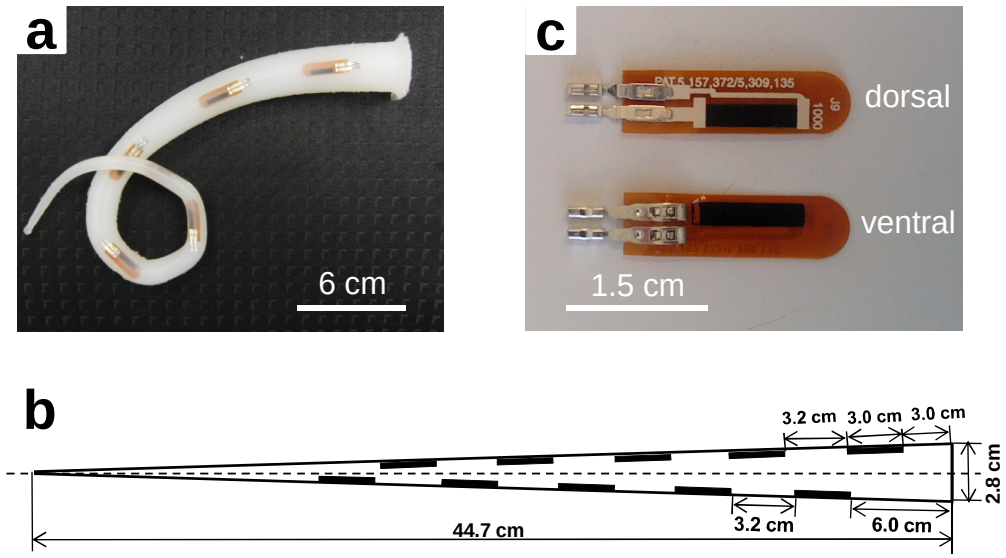
Short-term memory in soft body dynamics

Supplementary Information

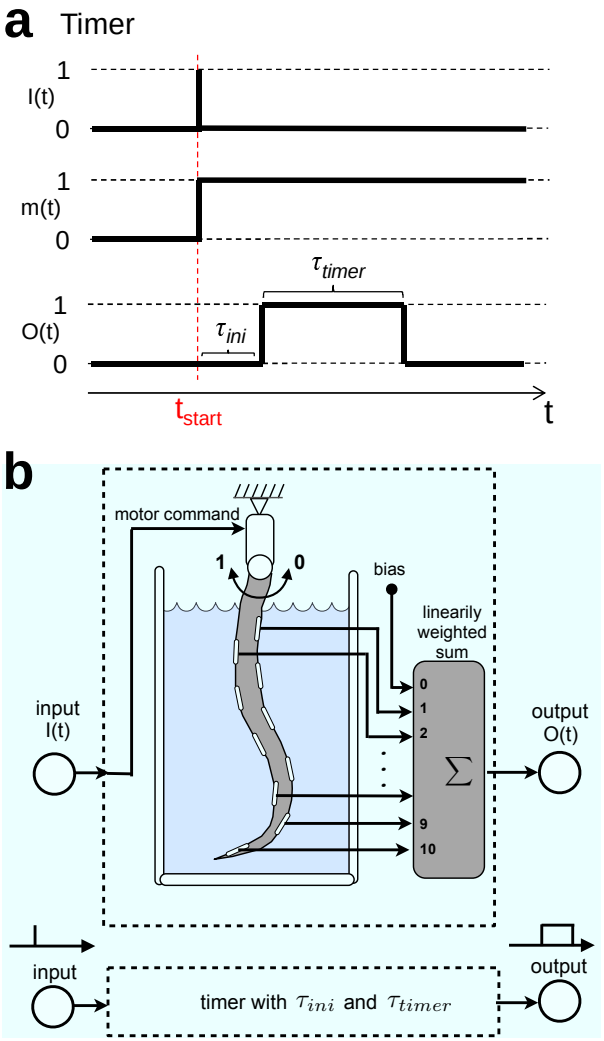
Kohei Nakajima *, Tao Li , Helmut Hauser , Fumiya Iida , Rolf Pfeifer

1 Department of Mechanical and Process Engineering, ETH Zurich, Leonhardstrasse 27, 8092 Zurich, Switzerland.
2 Department of Informatics, University of Zurich, Andreasstrasse 15, 8050 Zurich, Switzerland.

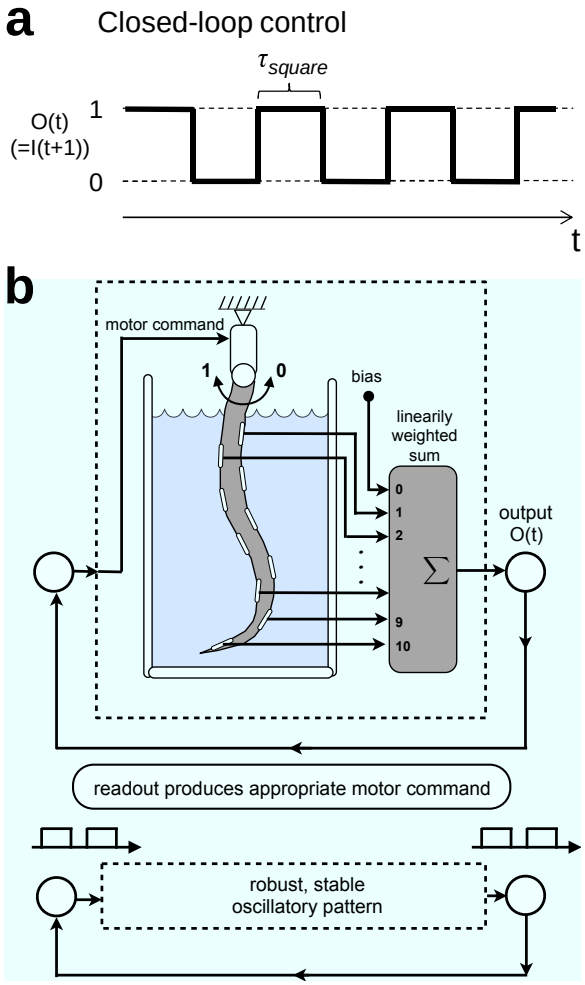
Supplementary Figures



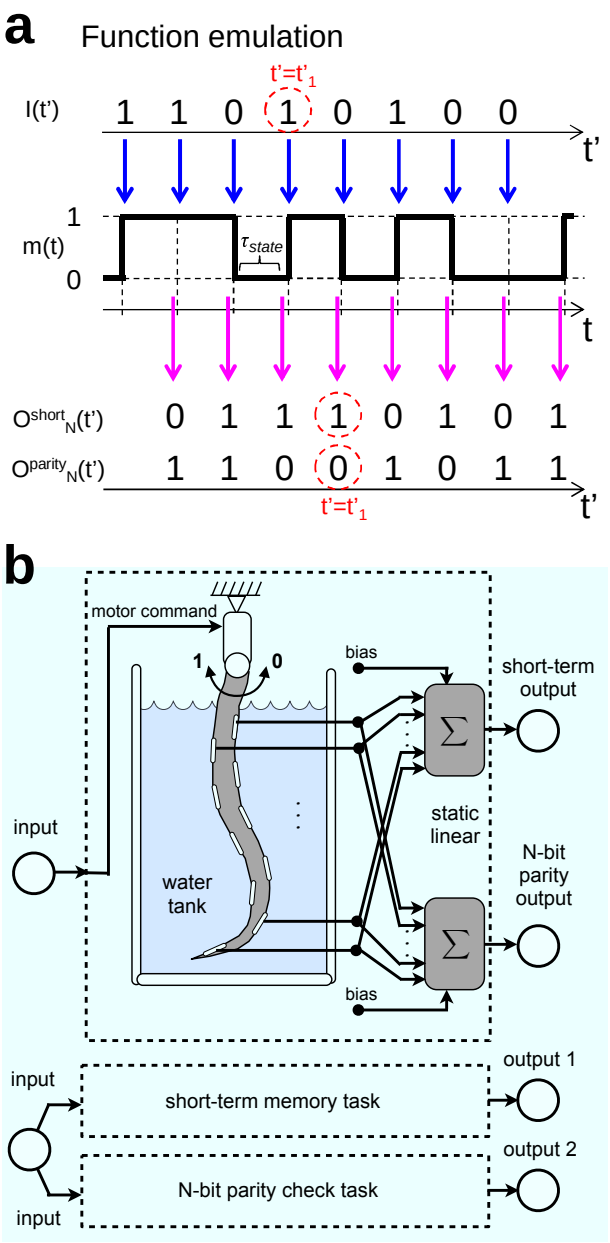
Supplementary Figure S1: Soft silicone arm and bend sensors. (a) A soft silicone arm with 10 embedded bend sensors. (b) Schematics showing the alignment of the bend sensors in the arm. The sensors are aligned parallel to the arm surface with an equal distance of 3.2 cm between them. There is a thin layer of silicone of about 0.1 cm covering the sensors. (c) The bend sensors. Both dorsal and ventral sides are shown.



Supplementary Figure S2: Experimental procedure for the timer task. (a) Schematics showing the I/O relation with respect to the temporal axis. Triggered by the input at $_{start}$, the motor command switches from 0 to 1. By exploiting the sensory time series resulting from the soft body dynamics, the system should output a pulse with $_{timer}$ timesteps in length after $_{ini}$ timesteps from timestep $_{start}$. (b) Schematics showing the generation of the system output. The system output is generated by thresholding the weighted sum of the corresponding sensory time series.



Supplementary Figure S3: Experimental procedure for the closed-loop control task. (a) Schematics showing the I/O relation with respect to the temporal axis. The target motor command is a square function with $_{square}$. (b) Schematics showing the generation of the system output in closed-loop control task. The closed-loop control is realized by feeding back the generated output to the input at the next timestep.



Supplementary Figure S4: Experimental procedure for the function emulation tasks. (a) Schematics showing the I/O relation with respect to the temporal axis. The time scale defined for the I/O relation is ′. The input symbol is provided to the system for each $_{state}$ timestep. The corresponding sensory time series, $_i(\ ')$, is at timestep $(\ ' +\ ) * \ _{state} -\ \ (\ '$ is presented as a reference to show the I/O relation). (b) Schematics showing the generation of the system output for function emulation tasks. The system outputs are generated with a multitasking scheme, i.e., the same soft body is employed to carry out two different tasks at the same time.

Supplementary Movies
(Captions)

Supplementary Movie 1: Example showing the basic arm motion, the corresponding sensory responses, and the system performance for the timer task. The arm is initially set to $_{right}$; when triggered by the motor command switched from 0 to 1, the arm moves from $_{right}$ to $_{left}$. The corresponding sensorimotor time series are shown with the arm behavior. An example of the performance of the embedded timer is also shown. By exploiting the same sensory time series, the system emulates a timer for given $_{ini}$ and $_{timer}$ with already-trained readouts. The system outputs for the cases of $_{timer} =$, , , , , and are shown with the target outputs, where $_{ini}$ is fixed to 9. The average system outputs over 25 trials are also shown as a reference for each condition.

Supplementary Movie 2: Example showing the system performance for the closed-loop control task. First part of the video shows the system performance for the closed-loop control task for square wave functions with $_{square} =$, , , , and driven with already-trained readouts. Arm motions as well as corresponding sensorimotor time series and the target motor command are shown for each $_{square}$. One can confirm that as $_{square}$ increases, the system performance decreases. The second part of the video shows the system performance for the same closed-loop control task with a manual mechanical disturbance to the arm. Arm motions as well as corresponding sensorimotor time series and the target motor command are shown for each case. When $_{square} =$, we can see that the system output returns to nominal time series that emulates the given square function after the perturbation, which shows that the system is robust against external perturbation. When $_{square}$ increases, one can confirm that the robustness of the system decreases, and the arm stops occasionally at $_{right}$ or $_{left}$.

Supplementary Movie 3: Example showing the system performance for the function emulation tasks. The system performance for the short-term memory task and the N-bit parity check task during the evaluation phase are shown for $_{state} =$, , and . For each $_{state}$, the system outputs for $=$, , and are shown together with the target output. Arm motions as well as corresponding sensorimotor time series are shown for each case.

Supplementary Methods

Training the readout weights using logistic regression

In this manuscript, we provide a brief overview of logistic regression and how it is used to train the readouts from the sensor to produce the desired output. Detailed information on the logistic regression can be found in [26]. As is described in the main text and in Methods section, our system output takes binary states for each task by thresholding the value obtained by summing up the linearly weighted sensory values. This is basically a two-class classification of the sensory values, which can be appropriately dealt with the logistic regression model. Following the notation used in the main text, we have 11 sensory time series $_i(\ )(\ =\ ,\ ,...,\ )$ and corresponding 11 linear and static readout weights $_i$ $(\ =\ ,\ ,...,\ )$. Note that, as explained in the main text and in Methods section, a unit expressing time for the function emulation tasks was $'$ due to an input symbol introduced to have a specific duration of time, i.e., $_{state}$. In the following descriptions, we keep using for the general case, but one can replace with $'$ for the function emulation tasks. We also introduce the vectors $\mathbf{s_t} =\ (\ ),\ (\ ),...,\ (\ )$ and $\mathbf{w} =\ ,\ ,...,$ for descriptive purposes. Our aim here is to optimize $\mathbf{w}$ by using corresponding $_{train}$ data pairs, $\mathbf{s_t}$ and $_{target}(\ )$, collected in the training phase in each task. For example, in the first task of constructing a timer, we had 25 trials, which had 250 timesteps each, as training data. This results in $_{train} =\ \times\ =$ data pairs. In the closed-loop control task and the function emulation tasks, $_{train} =$ data pairs were used for training (see Methods section).

Based on [26], we start by introducing a posterior probability for classes and , which correspond to the output states 1 and 0, respectively,

$$(\ \ |\mathbf{s_t}) =\ (\mathbf{s_t}) =\ (\mathbf{w}\ \ \mathbf{s_t}),$$
$$(\ \ |\mathbf{s_t}) =\ -\ (\ \ |\mathbf{s_t}),$$

where $(\cdot)$ is the logistic sigmoid function. We now make use of the maximum likelihood method to determine the optimal weights. For a data set $(\mathbf{s_t},\ _{target}(\ ))$, where $_{target}(\ )$, and $=\ ,...,\ _{train}$, the likelihood function can be expressed as

$$(\mathbf{O_{target}}|\mathbf{w}) = \prod_t\ \ ^{O}_{t}\ \ ^{t}(\ -\ _t)^{\ -O}\ \ ^{t},$$

where $\mathbf{O_{target}} =\ _{target}(\ ),\ _{target}(\ ),...,\ _{target}(\ _{train})$ and $_t =\ (\ \ |\mathbf{s_t})$. The error function of this likelihood can be expressed as

$$(\mathbf{w}) = -\ \ (\mathbf{O_{target}}|\mathbf{w}) = -\sum_t\ \ _{target}(\ )\ \ _t + (\ -\ _{target}(\ ))\ \ (\ -\ _t)\ ,$$

where $_t = \ (\ _t)$ and $_t = \mathbf{w}\ \ \mathbf{s_t}$. Taking the gradient and Hessian of this error function, we obtain

$$(\mathbf{w}) = \sum_t^T (\ _t - \ _{target}(\ ))\mathbf{s_t} = \mathbf{S}\ \ (\mathbf{y} - \mathbf{O_{target}}),$$

$$\mathbf{H} = \ \ (\mathbf{w}) = \sum_t^T \ _t(\ - \ _t)\mathbf{s_t}\mathbf{s_t}\ \ = \mathbf{S}\ \ \mathbf{RS},$$

where $\mathbf{y} = \ \ , \ \ , ..., \ _T$ and $\mathbf{S}$ is a matrix expressed as

$$\mathbf{S} = \begin{matrix} \mathbf{s_1} \\ \vdots \\ \mathbf{s_{T_{train}}} \end{matrix} = \begin{matrix} (\ ) & \cdots & (\ ) \\ \vdots & \ddots & \vdots \\ (\ _{train}) & \cdots & (\ _{train}) \end{matrix}.$$

Additionally, we introduce the $_{train} \times \ _{train}$ diagonal matrix $\mathbf{R}$ with elements, $_{tt} = \ _t(\ - \ _t)$. To minimize the function $(\mathbf{w})$, we used the iterative reweighted least squares method. The Newton-Raphson update formula for the logistic regression model is expressed as

$$\begin{aligned} \mathbf{w}^{\ new} &= \mathbf{w}^{\ old} - \mathbf{H}^{-}\ \ \ (\mathbf{w}) \\ &= \mathbf{w}^{\ old} - (\mathbf{S}\ \ \mathbf{RS})^{-}\ \mathbf{S}\ \ (\mathbf{y} - \mathbf{O_{target}}) \\ &= (\mathbf{S}\ \ \mathbf{RS})^{-}\ \ \mathbf{S}\ \ \mathbf{RSw}^{\ old} - \mathbf{S}\ \ (\mathbf{y} - \mathbf{O_{target}}) \\ &= (\mathbf{S}\ \ \mathbf{RS})^{-}\ \mathbf{S}\ \ \mathbf{Rz} \end{aligned}$$

where $\mathbf{z}$ is an $_{train}$ -dimensional vector with elements

$$\mathbf{z} = \mathbf{Sw}^{\ old} - \mathbf{R}^{-}\ (\mathbf{y} - \mathbf{O_{target}}).$$

We apply this procedure iteratively, each time using the new weight vector $\mathbf{w}$ to compute an updated weighing matrix $\mathbf{R}$, until $\frac{\|\mathbf{w}\ \ \ -\mathbf{w}\ \ \ \|}{\|\mathbf{w}\ \ \ \|}$ is less than 0.1 in all our experiments.

Appendix G

Curriculum Vitae

Curriculum Vitae

Last name:	Li
First name:	Tao
Data of birth:	October 25, 1983
Nationality:	Chinese
Marital status:	Married

Education

02/2009 – 08/2013	Ph.D. candidate and research assistant at the Artificial Intelligence Laboratory, Department of Informatics, University of Zurich, Switzerland Ph.D. thesis: Learning from the Octopus: Sensorimotor Control of Octopus-Inspired Soft Robots
09/2005 – 12/2007	Master in Mechanical Engineering, Department of Mechanical and Energy Engineering, Zhejiang University, China Master thesis: Study on the Piston Ring Rotary Seal and its Sealing Properties
09/2001 – 07/2005	Bachelor in Mechanical Engineering, Department of Mechanical Engineering, Huazhong University of Science & Technology, China